The ASAD atmospheric chemistry integration package and chemical reaction database

USER GUIDE

Release 2.0 : May 1997

Glenn Carver, Paul D. Brown and Oliver Wild

Centre for Atmospheric Science
Chemistry Department
Cambridge University
Lensfield Road, Cambridge
CB2 1EW, UK

Table of Contents

1 Introduction 1
- 1.1 How to get ASAD 1
- 1.2 What's new in this version 3
- 1.3 Compiling ASAD 3
  - 1.3.1 Compiler options 4

2 The chch.d file 5

3 Select and the ASAD reaction database 8
- 3.1 Ratefiles 8
- 3.2 Running SELECT 11
- 3.3 The info.doc file 12
- 3.4 Editing the ratefiles and chch.d before running ASAD 12
- 3.5 Updating the rates from new master ratefiles 13
- 3.6 Fractional products 13

4 How ASAD works 15
- 4.1 Basic principles 15
- 4.2 Input ratefiles and chch.d 17
- 4.3 ASAD parameters 17
  - 4.3.1 The PEPS variable 18
- 4.4 Call to subroutine CINIT 19
- 4.5 Call to subroutine CDRIVE 20
- 4.6 Chemical families in ASAD 20
  - 4.6.1 Calculating the family ratios 22
- 4.7 Time integration schemes 23

5 User supplied subroutines 25
- 5.1 The rate array, rk. 25
- 5.2 Subroutines inphot & photol 26
- 5.3 Subroutines inemit & emissn 26
- 5.4 Subroutines inddep, inwdep, drydep & wetdep 26
- 5.5 Subroutine hetero 27
- 5.6 Adding constant field types ('CF') 27

6 Editing ASAD 29

7 Setting up an atmospheric chemistry model: A step-by-step example 30
- 7.1 Stage 1. Selecting a chemistry. 30
- 7.2 Stage 2. Running SELECT 31
- 7.3 Stage 3. Editing the chch.d file. 33
- 7.4 Stage 4. Editing the ratefiles. 34
- 7.5 Stage 5. Setting up the atmospheric model. 36
- 7.6 Stage 6. Setting ASAD parameters. 37
- 7.7 Stage 7. Model-ASAD interface: cinit. 38
- 7.8 Stage 8. Model-ASAD interface: cdrive. 39
- 7.9 Stage 9. Supplying a photolysis scheme. 39
- 7.10 Stage 10. Supplying a heterogeneous reaction scheme. 40

7.11 Stage 11. Supplying emissions and deposition rates. 40
7.11.1 How to handle emissions and families 41
8 Acknowledgements 43
9 References 44
Appendicies 45
A.1 License information 45
A.2 Utilities supplied with ASAD 46
A.3 Computational efficiency and implementing ASAD on multiple processors ..47
A.4 Programming standard 48
A.5 Species naming convention 49
A.7 Known bugs and problems 51

1 Introduction

ASAD (A Self-contained Atmospheric chemistry coDe) (see also Carver et al. 1997)[1] is an atmospheric chemistry package designed for use in any atmospheric model using any number of species and reactions. Species can be treated individually or as members of a family; they can even be allowed to switch between the two states depending on their lifetime. ASAD also supports other approximations commonly used in atmospheric chemistry models; such as steady state species and the use of fractional products for dealing with long reaction chains.

ASAD includes treatment of bimolecular, termolecular, photolysis and heterogeneous reactions, emissions and wet and dry deposition to the surface. Several published time integration methods are available, both stiff and non-stiff methods. The user is free to choose whichever scheme they feel best suits their problem. No coding or programming is required to change integration scheme.

ASAD is easy to use. The user simply supplies a file with a list of species, along with a few option flags which are used to customise the chemistry to her/his needs. First, this species set is used as input to the program SELECT. This program selects a subset of reactions from a master reaction database that contain the chosen species. The output ratefiles, one for each reaction type, either in their raw form or after editing, together with the species set are subsequently used as input to ASAD. The only additional software that the user must supply are:

- a model-ASAD interface routine for the calling atmospheric model.
- if required, a photolysis scheme
- if required, code to calculate heterogeneous reaction rates
- if required, code to calculate rates for emissions and wet and dry depositions
- if required, code to set field constants

Subroutines to perform these tasks are not supplied with ASAD because they are either model dependent or they must be tailored to suit the chosen species or experiment. We describe these requirements in more detail later in this userguide.

In general, it should not be necessary to alter ASAD, however, we have endeavoured to make the layout of the code as logical as possible so that editing it is relatively straightforward.

1.1 How to get ASAD

Before you get the ASAD package, you should read the license information in the appendix. This covers your rights to use and redistribute the software and our responsibilities. In particular, we ask that you do not redistribute the package without informing us, you also inform us of improvements or changes which might be of

1. The user should read the paper in conjunction with this user guide. The paper gives some information not covered in this user guide.

benefit to everyone so that we can include them in the original copy and that you respect the development effort put into ASAD by appropriate acknowledgement in any published papers arising as a result of the use of ASAD. We want to stress however, that ASAD is freely available and that we encourage the use of ASAD in other groups (it's partly why we wrote it) and will endeavour to support the package as far as our resources allow.

The best place to start is the ASAD World-Wide Web pages at:

http://www.atm.ch.cam.ac.uk/acmsu/asad/

These pages contain a description of ASAD, how to get the package, the user guide in hypertext format so that it can be read on the WWW plus details of contacts for the package.

From Version 2.0, the complete ASAD package will be available from the Computer Physics Communication Program Library (http://www.cs.qub.ac.uk/cpc/); email cpc@qub.ac.uk. The ASAD web page contains details of how to get the software in case of problems retrieving the software from the CPC program library. We are happy to help in case of difficulty.

To obtain the individual files after transferring the tarfile, type the following UNIX commands:

```
% mkdir asad
% uncompress asad.tar.Z
% tar -xvf asad.tar
```

The distribution will unpack itself in the current directory with the following files and directories:

README	List of files and directories.
BUGS	List of known problems and bugs.
VERSIONS	Contains a complete version history of the source code.
makefile.sample	A sample makefile with which to compile ASAD.
LICENSE	Description of license information for ASAD.
docs/	Contains the user guide in PostScript form.
nupdate/	Empty. Used in makefile.sample to create CRAY NUPDATE source code.
rates/	Holds the master ratefiles from the UGAMP/ACMSU reactions database.
src/	Holds all of the ASAD source code.
utils/	Some utility programs.
tests/	Holds all the test programs which you should run first to verify the installation is correct.

In the rates/ subdirectory you will find the following files:

bimol.d	ASAD master ratefile of bimolecular reactions.

trimol.d	ASAD master ratefile of termolecular and unimolecular reactions.
photol.d	ASAD master ratefile of photolysis reactions.
hetero.d	ASAD master ratefile of heterogeneous reactions.
species.d	A list of all species appearing in the ASAD reaction database.

In the utils/ subdirectory you will find the following files:

sel_job	UNIX job script for running SELECT.
select.f	Source code for the SELECT program.
mergerate.f	Source code for the MERGERATE program to updates rates from the master ratefile.
f2up.c	C program to convert FORTRAN source code to NUPDATE source code.
renumb.f	Program for renumbering edited ratefiles.

For more details on the programs in the utilities directory see the Appendix. The SELECT program is described below.

1.2 What's new in this version

Version 2.x of ASAD has seen some major changes to the code from version 1.0. There has been a lot of rewriting of the internal workings of ASAD, in particular the subroutines that account for most of the CPU time. In brief, the main changes are:

- Major rewrites to most ASAD subroutines to achieve large speed-ups; factors of 5 or more on machines we have access to.
- New features such as the use of fractional products in specifying reactions.
- Additional time integration routines, particularly for stiff problems.
- Improved format for chch.d file and new constant field type.
- Ratefiles now support the use of comment lines denoted by #.
- More structured calling interface.

Like all computer programs, there will be bugs or restrictions on the use of the code. See section A.7 in the Appendix for more details.

1.3 Compiling ASAD

How you use ASAD in your programs is largely up to you. In our group, each user has a read-only copy of the ASAD code that they compile with their models. On the Cray however, we find it easier to create a library of the ASAD code.

There is a sample makefile in the ASAD distribution (called makefile.sample) which you can use as a template to compiling the code.

On the Cray, we use the source code control system called UPDATE (or the new version called NUPDATE). The makefile.sample can be used to format UPDATE source from the Fortran source in the src/ subdirectory (e.g. type ‘make -f makefile.sample update’). UGAMP users should use the update library in /home/j90/kd/chem/lib/ on the Cray.

ASAD has been run on many different hardware architectures. It is written in standard FORTRAN77 to ensure it’s as portable as possible. The only non-standard aspect of the code is the use of variable names longer than 6 characters. We don’t believe this will be a problem for modern day FORTRAN compilers. Some of the utility programs also use the DO-ENDDO construct, but the ASAD code itself uses DO-CONTINUE.

At the time of writing this, ASAD has been successfully used on: Cray YMP and Cray J90 (running Unicos), Sun SPARCstation (10,20,Ultra) running Solaris1 and 2, Silicon Graphics Indigo running IRIX 5, IBM RS6000 running AIX, Apple MacIntosh (running MacOS 7.5) and Pentium and Pentium Pro PCs running Linux. The only non-Unix machine we have run it on is the Apple Mac. Basically, any machine with a half decent FORTRAN compiler should be able to run to run ASAD. All the I/O that ASAD does is completely standard.

1.3.1 Compiler options

In this section we briefly mention a few compiler options you might need to use. We strongly recommend that if you are running ASAD on a 32-bit machine, where the default word length is 32bits, that you promote FORTRAN single precision variables and constants to double precision through the use of compiler options. On Cray vector machines of course, word length is 64bits to variables are by default 64 bits. However, for other machines we recommend 64bits to avoid the possibility of the loss of precision. The ASAD code was not written to use DOUBLE PRECISION deliberately because we knew it would have to run on 32 bit and 64 bit word machines and DOUBLE PRECISION on the Cray would be 128bits which is not necessary.

On our Suns and Silicon Graphics workstations we use the ‘-r8’ option to promote all real and integer variables and constants that have an assumed size to 64 bits. Although we don’t have access to an RS6000 we’ve been told that either the ‘-qrealsize=8’ or ‘-qautodbl=DBL4’ options will work. On the Apple Mac, we use the LS FORTRAN compiler from Fortner Research (you can find them on the WWW). This compiler supports the ‘-Cray’ option which promotes variables to double precision.

The authors would be interested to hear of any other options on other hardware platforms that ASAD needs.

2 The *chch.d* file

ASAD has been designed to allow either the use of the “family approach” commonly adopted in atmospheric chemistry, or be used with all species integrated separately. We will assume for the purposes of this user guide that the reader is familiar with the family method of solving the chemical equations (Austin 1991).

It is necessary at this stage to distinguish between the tracers that are held in the atmospheric model, which can be either families or individual species, and the species that take part in the chemistry. We will often refer to the former as “tracers” or “families” and the latter as “species”.

The ASAD user must supply a file named chch.d (for ‘chosen chemistry’) containing a list of those species that take part in the chemistry. This file is used by both SELECT and ASAD. In addition to the species names, it must contain three flags for each species that determine how that species is treated within ASAD. The values set for these flags and the order of the species are unimportant when chch.d is used in SELECT, but they are critical for the correct functioning of ASAD. When designing a chch.d file for use with SELECT, it is vitally important to ensure that all the species involved in the chemistry are listed, even those that will not be held in the model or are members of a family. After running SELECT, it is possible to edit chch.d prior to running ASAD, and we will discuss possible reasons for doing so in section 3.4. The other important point is that the species must be identified by the names given to them in the ASAD reaction database. This is because of the need to match character strings during the running of both SELECT and ASAD. Species names are restricted to 10 characters. Oliver Wild of the University of Cambridge has devised a nomenclature for species whose commonly used names or chemical formulae are longer than this imposed limit (see Appendix). A list of species currently featured in the ASAD reaction database is given in the file species.d, which is included with the package.

The format of the chch.d file is relatively straightforward. FORTRAN list directed I/O is used to read this file to precise spacing is not important. An example section of such a file is shown below:

Table 1: chch.d format

Species no.	Species name	No. of odd atoms	Treatment	Family name
1	‘O(1D)’	1	‘FM’	‘Ox’
2	‘O(3P)’	1	‘FM’	‘Ox’
3	‘O3’	1	‘FM’	‘Ox’
4	‘Cl’	1	‘FM’	‘ClOx’
5	‘Cl2O2’	2	‘FM’	‘ClOx’
6	‘OClO’	1	‘FT’	‘ClOx’

Table 1: chch.d format

Species no.	Species name	No. of odd atoms	Treatment	Family name
7	‘ClO’	1	‘FM’	‘ClOx’
8	‘OH’	1	‘SS’	‘ ‘
9	‘CH4’	1	‘TR’	‘ ‘
10	‘N2O’	1	‘TR’	‘ ‘
11	‘MeOOH’	1	‘TR’	‘ ‘
12	‘CO2’	1	‘CT’	‘ ‘
13	‘N2’	1	‘CT’	‘ ‘

The integer in the first column is the index used to identify each species. The second column contains the species names, in ASAD format. The inverted commas are necessary because of the use of list directed input in SELECT and ASAD. However, this does mean that the number of spaces between each entry in the file is unimportant. Column 3 must indicate the number of “odd atoms” of the appropriate family type contained in that species. For example, for the chlorine family, ClO_x, the species Cl_2O_2 contributes two odd atoms of chlorine to the family because Cl_2O_2 photolyses to give Cl + Cl + O_2. On the other hand O_3 only contributes one odd atom to the O_x family. The no. of odd atoms each species contributes to a family is not the same as counting the no. of atoms in that species! The no. of odd atoms is only really important for species that are members of families. The 2 character code in column 4 is used to specify how the species should be treated in ASAD. The following options are available:

TR The species is integrated separately as an independent chemical tracer (not part of a family, not constant or in steady state e.g. N_2O).

FM The species is to be treated as a member of the family specified in column 5. This family is one of the tracers passed to ASAD by the calling atmospheric model.

FT The species is to be integrated separately unless its lifetime during any one particular model timestep is less than half the chemistry timestep (the two timesteps may be different - see section 4.1), in which case it is to be treated as though a member of the family specified in column 5. Both the species and its associated family will be tracers in the model (n.b. the concentration of the family stored in the model should not include that of this species).

SS The species is neither stored in the model or a part of a family but, for the purposes of the chemistry, can be assumed to be in steady-state.

CT The species is not stored in the calling atmospheric model but, for the purposes of the chemistry, is assumed to have a fixed

volume mixing ratio throughout the portion of the atmosphere under consideration. The species that may fall in to this category, and their concentrations, are: CO_2 (3.5 x 10-6), H_2 (5.0 x 10-7), O_2 (0.20945) and N_2 (0.78084). Other species and/ or concentrations may be added to this list by editing ASAD (see sections 6 and 7) as these concentration are ‘hard-wired’ into the code.

CF Like the ‘CT’ species type, this species is also treated as a constant, but one which can have a different value at each point in the spatial domain. In order to set this constant field, a subroutine which the user must supply, INICNT, is called at the appropriate place in ASAD.

Note that only families (options FM and FT) and species integrated separately (TR and FT) are passed between the calling model and ASAD. Family members (FM), steady state species (SS) and constant species (CT and CF) are not passed.

Column 5 contains the name of the family, where appropriate, to which the species belongs. If the species does not belong to a family, it is necessary to enter a blank character enclosed in inverted commas because of the use of list directed input. Family names must be 10 characters or less, and must not be the same as any of the species names. ASAD will always use the last named species of a particular family as the major species in that family. Therefore, it is critically important to order the family species in the chch.d file in the right way.

3 Select and the ASAD reaction database

3.1 Ratefiles

There are four master ratefiles in the ASAD reaction database, one for each category of reactions:

bimol.d	Bimolecular reactions.
trimol.d	Termolecular reactions and unimolecular.
photol.d	Photolysis reactions.
hetero.d	Heterogeneous reactions.

SELECT is designed to scan these master ratefiles or alternative master ratefiles and select only those reactions relevant to the chosen species listed in the chch.d file, and write them to output files. The output ratefiles are then used as input to ASAD.

The reactions have been compiled mainly from the IUPAC Supplement IV assessment (Atkinson et al. 1992) and the JPL reaction handbook (DeMore et al. 1992). Reaction data in refereed papers have also been used on occasion for reactions that have not previously been measured or for new reactions. These are indicated in the comment field in the ratefile. In general however, we felt it was appropriate for the master ratefiles to adhere to carefully reviewed kinetic data. However, the user is free to create their own master ratefiles and adopt their own policy with regard to usage of published kinetic data. It is our intention to update the ASAD reaction database when new assessments are published (see the header lines in the ratefile for version information).

All ratefiles are in ASCII and use an easy to read format. In any ratefile all ASAD programs will treat a line beginning with a ‘#’ as a comment and ignore it. This is useful for instance for temporarily taking a reaction out of the chemistry scheme. Rather than delete the line in the ratefile, the reaction can simply be commented out.

The essential point to note here is that the ratefiles contain information about the reactants and products of a reaction and, in the case of bimolecular and trimolecular reactions, the parameters necessary to calculate the rate coefficient. The ratefiles are laid out in columns: the first is the reaction index, followed by the two reactants (a third body is assumed in termolecular reactions), the next three (two in the termolecular case) are the products and the remaining columns contain the parameters needed to calculate the rate coefficients (these are dummy arguments for photolysis and heterogeneous reactions). A question mark ‘?’ in any of the product columns indicates that the products of the reaction are unknown.

The master ratefiles may not always contain all of the reactions of interest to a particular user. Customising ratefiles to meet the specific needs of a user is straightforward. This can be done at one of two stages. If it is likely that the additional set of reactions will be used on more than one occasion, then it makes sense to add those reactions to a copy of the master ratefile. On the other hand, if it is simply a case of adding a single reaction or changing a particular rate coefficient for a certain experiment, the output ratefiles should be edited (see sections 3.4 and 7). Since the list of photolysis and heterogeneous reactions may be highly dependent on the user

supplied schemes, we recommend that users always define their own photolysis and heterogeneous master ratefiles. These ratefiles must adopt the same formats as those in the ASAD database.

We welcome any additions to the database held at Cambridge. Note we only use peer reviewed published rate data.

3.1.1 Bimolecular ratefile

A few lines from the bimolecular ratefile are shown in Table 2.

Table 2: Example bimolecular ratefile

8	Br	O3	BrO	O2		1.70E-11	0.00	800.0	
9	Br	OClO	BrO	ClO		2.60E-11	0.00	1300.0	
10	BrO	BrO	Br	Br	O2	9.20E-13	0.00	250.0	C
11	BrO	BrO	Br2	O2		1.80E-13	0.00	250.0	C

The ratefiles are laid out in columns; the first is the reaction number in the file, the next 2 contain the reactants, the next 3 columns contain the products and the subsequent 3 columns contain the parameters needed to calculate the rate coefficient for the reaction. The last column is a character string used for comments, or to refer to notes at the bottom of the ratefile. For example, to list references for those reactions taken from sources other than Atkinson et al. (1992) and DeMore et al. (1992) or to denote kinetic data which is only known to an upper limit. Blank entries are acceptable where a reaction does not have enough products. If it is necessary to have more than 3 products, the user must alter a parameter in the ASAD code (see paramc.h) and reformat the ratefiles accordingly. A utility tool is provided for this purpose. ASAD uses a fixed format FORTRAN read and write for these ratefiles so positioning of the columns is important. Species names must be no longer than 10 characters and must be a recognised species name in the ASAD nomenclature system (see Appendix).

For the bimolecular ratefile, columns 7 (k_0), 8 (α) & 9 (β) are used to compute the rate coefficient, k, from the Arrhenius expression

$$k = k_0\left(\frac{T}{300}\right)^{\alpha}\exp\left(\frac{-\beta}{T}\right) \qquad \text{(EQ 1)}$$

in units of $cm^{-3}s^{-1}$, where T is the temperature in Kelvin. The subroutine BIMOL is used to compute all the bimolecular rate coefficients.

There are exceptions to this general formula. For example, the reaction between CO and OH is pressure dependent. However, there are only a handful of reactions that do not use Eq. 1. Rather than alter the ratefile format, these reactions are flagged in column 10 and notes on how to compute the reaction rate are given at the end of the ratefile. For such reactions listed in the ASAD reaction database, code specific to that reaction has been included in BIMOL to give the correct rate coefficient.

A question mark '?' in any of the product columns indicates that the products are unknown even though a measurement for the reaction rate exists (this applies to all ratefiles). We will discuss the significance of these reactions in choosing a reaction subset in the following section.

There are reactions which have two or more possible branches. Each branch must have a separate entry in the ratefile. Where the branching ratio is unknown, we have assumed that the branches occur at equal rates i.e. 50:50 ratio in the case of two branches, 33:33:34 for three branches and so on. This is accomplished by dividing the total measured k_0 parameter by the number of branches.

3.1.2 Termolecular ratefile

Although called the termolecular ratefile we also include unimolecular reactions in this file as unimolecular and termolecular reactions can be thought of as consisting of the same elementary processes and can exhibit varying order depending on pressure (see Wayne, 1991).The termolecular file follows a similar format to the bimolecular ratefile and an example is shown in Table 3.

Table 3: Example termolecular ratefile.

N	Reactants		Products		f	k_1	α_1	β_1	k_2	α_2	β_2
1	BrO	NO2	BrONO2	m	327.00	4.70E-31	-3.10	0.0	1.70E-11	-0.60	0.0
2	Cl2O2	m	ClO	ClO	0.60	1.35E-05	1.00	870.0	1.80E+15	0.00	8450.0

For all reactions a third body is assumed. For unimolecular reactions, the character 'm' appears in place of the second reactant.

The number of parameters needed to calculate the rate coefficients for termolecular reactions is greater than in the bimolecular case. The temperature and pressure dependent rate coefficient, k, is found from the rate constants for the low pressure, k_0, and high pressure, k_∞, limits by

$$k = \left(\frac{k_0[\mathrm{M}]}{1 + k_0[\mathrm{M}]/k_\infty}\right) F_c^{\left(1 + \left[\log 10 \frac{k_0[\mathrm{M}]}{k_\infty}\right]^2\right)^{-1}} \qquad \text{(EQ 2)}$$

in units of $cm^{-6}s^{-1}$ where [M] is the concentration of the third body (molecules cm^{-3}) (using the notation of Atkinson et al. 1992). Typically, F_c is reported as a constant value, however it sometimes has a significant temperature dependence. To include this we use the following approach. If f is the parameter in column 6, when this value is less than 1, $F_c = f$ otherwise, $F_c = \exp(-T/f)$ where T is the temperature (in Kelvin) (see Atkinson et al. (1992), for more discussion).

The low pressure rate constant, which is also first order in pressure is found from

$$k_0 = k_1\left(\frac{T}{300}\right)^{\alpha_1}\exp\left(\frac{-\beta_1}{T}\right) \quad \text{(EQ 3)}$$

where k_1, α_1 and β_1 are the parameters given in columns 7-9 of the ratefile (Table 3). Similarly, the high pressure limit, k_∞, is found from

$$k_\infty = k_2\left(\frac{T}{300}\right)^{\alpha_2}\exp\left(\frac{-\beta_2}{T}\right) \quad \text{(EQ 4)}$$

where k_2, α_2 and β_2 are the parameters given in columns 10-12 of the ratefile.

One complication is that the low pressure limit, k_0, can sometimes be reported as dependent on the nature of [M] i.e. a different rate is given for N_2 and O_2. If only the k_1 parameter is different, then we take the weighted mean of the specified values for k_1. However, if the exponential factors, α_1 , are different, we list the reaction as having two branches in the ratefile and scale the parameter k_1 by the proportion of [M] in air. In both cases, notes are added to the ratefile for such reactions.

3.1.3 Photolysis and heterogeneous ratefiles

The photolysis and heterogeneous ratefiles are similar to the bimolecular ratefile. The columns used to specify the reaction rate are unused in these master ratefiles. This is because it is impossible to define a format which suits all photolysis and heterogeneous schemes. These columns are available to the user for specifying parameters for each reaction as part of their scheme. ASAD will read all the columns in the ratefile and the parameters are available in COMMON blocks in the code.

3.2 Running SELECT

The UNIX job script *sel_job* is provided with the package and should be used to run SELECT. The job script is very straightforward. It contains a section like that below for each reaction category:

```
#-----------------------------------------------------------------------
echo bimolecular reactions...
cat > control << EOF
 bimol.d
37
.true. .true. .true.
EOF
selx < control
#-----------------------------------------------------------------------
```

The script runs the program once for each reaction type. If heterogeneous reactions are not required, then the appropriate section can simply be omitted. Within each section, there are three lines containing the option flags to be set by the user. The first of these is the name of the master ratefile (*bimol.d* in the above example). On the following line, the number of species contained in the *chch.d* file (37 in this case). The next line

must contain three logical flags. These should be set .true. if the user wants as output: (1) an “open” reaction set; (2) a “closed” reaction set; and (3) the difference between open and closed sets. An open reaction set is one where the selection of reactions is made purely on the basis that the reactants are members of the chosen species set. In a closed set, the products must also be included in *chch.d.* Both sets have their advantages and disadvantages. An open set of reactions will contain all the destruction reactions of a species, even if the products are either unknown or unimportant. However, without due care, this may lead to non-conservation. A closed reaction network will certainly conserve, but may be missing important loss mechanisms for some species. We recommend that the user carefully studies the difference between the open and closed reaction networks. We feel that the best course of action is to use the “open” ratefiles as the basis for the ratefiles used as input to ASAD, editing them to discard any unwanted reactions (see sections 3.4 and 7). The output ratefiles from SELECT are:

rat?.open
rat?.close
rat?.diff

where the wild card character ‘?’ is either ‘b’ for bimolecular, ‘t’ for termolecular, ‘j’ for photolysis or ‘h’ for heterogeneous reactions.

3.3 The info.doc file

In addition to the ratefiles, SELECT writes an information file, *info.doc*, giving details about how ASAD would treat the chemistry on the basis of the flags set in *chch.d*. This includes a list of the tracers/families in the order in which they must be initialised in the model using ASAD. The major species in each family are identified.

3.4 Editing the ratefiles and chch.d before running ASAD

After running SELECT, the user may wish to edit the ratefiles prior to using them in ASAD. It is possible to add or delete reactions, or even change the products of a particular reaction. The rate of a reaction can be changed simply by editing the parameters used to calculate its rate coefficient (see above for further details of how the rates are calculated). Remember also that a reaction can be commented out by placing a ‘#’ as the first character on the line. Don’t worry about the numbering of reactions (first column), ASAD doesn’t actually use this, although the utility programs do.

As well as the need to study the differences between the open and closed set of ratefiles, there are several more reasons why they may have to be modified before being used by ASAD. For instance, if a reaction is included which has unknown products but is generally considered to be an important loss mechanism for one of its reactants, the user can make an assumption about the products formed. The user may also wish to exclude those reactions where the measured rate is listed as an upper limit only (indicated in the comment column in the ratefile) since their inclusion may introduce unrealistic effects. Or it may be desirable to retain them in the final ratefiles

in case new kinetic data subsequently becomes available. In this case, the reactions can effectively be turned off by setting all the rate coefficient parameters (columns 7 to 9) to zero. The user may also wish to attribute a fraction of the reaction to one of the specified products (see next section).

Earlier we stated that it was necessary to include all the species that might be involved in the chemistry in the file *chch.d* when running SELECT. However, there may be some species that react effectively instantaneously, and are of relatively little interest on their own. An example might be CH_3. Formed in the oxidation of CH_4, this species reacts rapidly with O_2 to form MeOO (CH_3O_2). It is possible to remove the explicit treatment of species like CH_3 in the chemistry by deleting them from the *chch.d* file, and by carefully editing the ratefiles. In this example, all reactions in which CH_3 is one of the reactants should be deleted, and in those reactions where CH_3 is one of the products, the string ‘CH3 ‘ should be replaced with ‘MeOO’. If the unwanted species reacts to form two other species, both must appear as products in the ratefile. Unfortunately, the number of reaction products is limited to three (two for trimolecular reactions), and so this kind of species reduction may not always be possible (although you could use one of the utility programs to reformat the ratefiles with 4 products).

A short program, RENUMB, is provided to renumber the ratefiles after they have been edited. This program can be run simply by compiling the file renumb.f and running the executable. The user is prompted for input and output filenames, and to say whether the format of the ratefiles is termolecular or not. Renumbering the ratefiles is not strictly necessary, the integer in the first column of both the ratefiles and the chch.d file is a dummy argument when the file is read by ASAD. However, doing so will help the user to identify the reactions by the indices used by ASAD. It should be remembered that deleting species from the *chch.d* file may affect the list of tracers/families written to the *info.doc* file by SELECT.

3.5 Updating the rates from new master ratefiles

From time to time, new versions of the ASAD reaction database will be released when new published data becomes available. A small Fortran program, called mergerate, has been written (available in the utils/ subdirectory) which takes the new reaction database and scans your own rate files for any reactions that are different. This is useful as some ratefiles can be quite large. See the Appendix for more information about mergerate or see the code itself.

3.6 Fractional products

Long and complex reaction chains can make ASAD expensive to run. Such long chains are frequently found in atmospheric hydrocarbon modelling and often approximated with reactions in which the final products are written as a fraction of the

production rate of the reaction. There are no fractional products used in the master ratefiles, but the user may include them in the bimolecular ratefile[1] produced by SELECT. For example, with a reaction

$$A + B \rightarrow C + D \qquad \text{with a rate} k \qquad \text{(EQ 5)}$$

the rate of change of C from this reaction is $dC/dt = fkAB$ where f is the fraction that contributes to the production of C. At present, f must be a constant. ASAD does not directly support any variation of f due to temperature or pressure. If this is necessary, the user must edit the ASAD source code.

Whilst strictly non-conservative, the use of fractional products can greatly reduce the number of reactions and therefore reduce the computational cost of the chemistry scheme. On a separate line under each of the products, a fraction expressed in FORTRAN ‘F’ format is placed. Instead of a number in the first (index) column of the ratefile, a ‘$’ character is used as a continuation marker. Positioning of these entries is important as ASAD uses fixed format I/O to read these ratefiles.

1. The use of fractional products is currently only supported for the bimolecular ratefile.

4 How ASAD works

We strongly suggest that the reader reads Carver et al. (1997) for extra detail not covered in this user guide.

4.1 Basic principles

An atmospheric chemistry model can be generally considered to consist of two main sections, a dynamical/transport component and a chemistry component. The dynamical part is responsible for the transport of the chemical species. The chemical part of the model is responsible for computing and integrating the chemical rates of change. To do this, it may use information from a photolysis scheme, possibly also a heterogeneous scheme (for example, reactions on cloud droplets), and perhaps also a surface scheme for emissions of source gases and wet and dry deposition schemes. Information can flow both ways between the packages. As indicated in the figure, ASAD is designed to take the place of the chemical part of the model and make use of user supplied photolysis, heterogeneous, deposition and emission schemes.

Consider the total rate of change of a species, *y*,

$$\left(\frac{dy}{dt}\right)_{\text{total}} = \left(\frac{dy}{dt}\right)_{\text{transport}} + \left(\frac{dy}{dt}\right)_{\text{chemistry}} + \left(\frac{dy}{dt}\right)_{\text{emis/dep}} \qquad \text{(EQ 6)}$$

The transport component of the atmospheric chemistry model will have the responsibility for the first term on the RHS of Eq. 6, whilst ASAD takes the responsibility for computing the remaining tendencies. ASAD computes the second term from tabulated reaction rates stored in input files and photolysis rates from the user supplied photolysis scheme, whilst the third term is computed by calling the other user supplied packages. The user may of course choose to use ASAD just for the second term and use their own method to integrate the deposition and emissions tendencies outside of ASAD.

ASAD therefore computes the total chemical rate of change of each species (the sum of the second and third terms on the RHS of Eq. 6) and integrates it forward in time. This rate of change of a species, *y*, is given by

$$\frac{dy}{dt} = P - Ly - ly^2 + E - Dy \qquad \text{(EQ 7)}$$

where P is the total chemical production of the species, L is the chemical loss rate, l is the loss rate when the species self-reacts, E is a production rate from emissions from the surface and D is the loss rate from dry and wet deposition. ASAD will compute the rate of production (P) and the total chemical loss rate (Ly - ly^2), from the information in the reaction rate input files. If the user wishes to include a source of emissions and loss through deposition, they must supply replacements for dummy subroutines in ASAD, otherwise these terms are zero and ignored by ASAD.

ASAD only integrates the tendencies for those tracers and families known to the calling model. For example, some species may be specified as constant or in steady state, in which case they are not integrated in time or known to the calling transport model, being defined within the ASAD subroutines only.

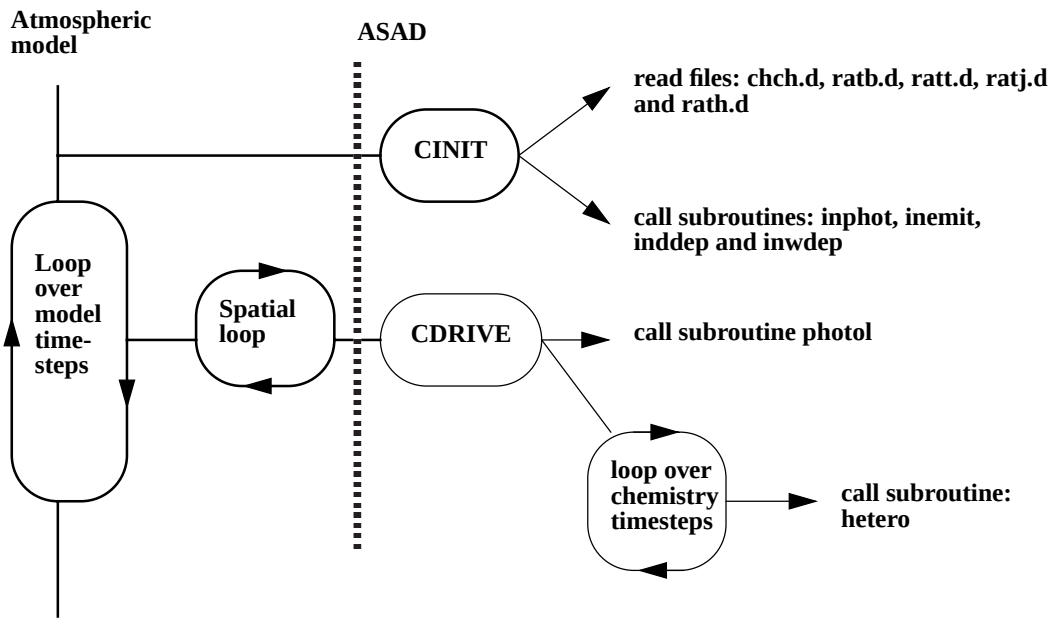
ASAD has been designed for use in any atmospheric model with any number of spatial dimensions. To accomplish this, ASAD uses arrays with one spatial dimension, thus enabling the code to be vectorized on machines with this capability. This spatial dimension can be either longitude, latitude or height or a combination (see the Appendix for a discussion about computational efficiency). The choice really depends on the structure of the calling model and the user defined photolysis scheme. Clearly, in box or trajectory models, the spatial index should always be set equal to 1.

The chemical lifetime of a species may be shorter than the dynamical timestep used in an atmospheric model. This is one reason why the “family approach” is often taken. However, even if families are used, the timestep used for transport may be too long to ensure accuracy and stability in the chemistry. Therefore, ASAD allows the chemistry to be integrated separately from the transport. For example, the transport model may use a ‘dynamical’ timestep of 30 minutes whilst the desired accuracy of the chemical solution may require a 10 minute ‘chemical’ timestep. ASAD would then perform 3 internal chemical sub-steps before returning to the calling model. During these sub-steps, the physical variables passed by the model, temperature and pressure, do not change.

A fast implicit integration scheme, called IMPACT (Carver and Stott 1997), has been built in to ASAD. IMPACT is the successor to the scheme developed by Stott and Harwood (1993). It is also possible to use other integration schemes by performing a few simple edits (see sections 6 and 7). The ASAD user is able to define the size of the chemistry timestep, and set key parameters for the integration scheme. The choice of values for these parameters is discussed fully in section 7. ASAD returns both the updated concentrations and the chemistry tendencies to the calling model. The chemistry tendencies can be either instantaneous values - if the chemistry was not integrated - or average values over the period of the model timestep. It can therefore accommodate models which process split the chemical integration and simply require the updated values, and also models that add the chemical tendencies to the transport tendencies in a non process split fashion.

In Fig. 1 below, we show the various interfaces between the calling atmospheric model and ASAD. The datafiles: chch.d, ratb.d, ratt.d, ratj.d and rath.d must be supplied by the user. The subroutines: INPHOT, INEMIT, INDDEP, INWDEP, PHOTOL and HETERO must be supplied by the user also. We will describe each of the interfaces in the sections that follow.

FIGURE 1. Flow diagram of ASAD code called from an atmospheric model



CINIT is called once per model run. It's job is to read the rate and species files and initialise constants and common variables. CDRIVE is the main subroutine that should be called each in model timestep. It controls the chemical substeps and the solution to the kinetic equations.

4.2 Input ratefiles and chch.d

ASAD expects as input the following files:

chch.d	The chosen chemistry file.
ratb.d	Bimolecular ratefile.
ratt.d	Trimolecular ratefile.
ratj.d	Photolysis ratefile.
rath.d	Heterogeneous ratefile.

All files must be in ASAD format. The heterogeneous ratefile, *rath.d*, need only be supplied if heterogeneous reactions are to be included (set by logical switch lhet - see section 4.4 below).

4.3 ASAD parameters

ASAD uses a number of parameters for dimensioning arrays. It is only possible to change these parameters by editing the parameters in the file *paramc.h*. The following parameters are those that the user might need to change when changing the chemistry scheme (no. of species or reactions):

jpctr	The number of advected tracers/families in the model.

jpspec	The number of species listed in the *chch.d* file, i.e. the number of species involved in the chemistry.
jpnl	The number of gridpoints passed to ASAD for the chemical tracers and families (used for the inner loop length).
jpbk	The number of bimolecular reactions.
jptk	The number of trimolecular reactions.
jppj	The number of photolysis reactions.
jphk	The number of heterogeneous reactions.

There are other parameters in **paramc**, however, in general it will not be necessary to change them. Most of them control array sizes. If an array that ASAD uses is too small, then a message will be printed out to indicate which parameter needs changing.

The parameters **ptol** and **ftol** control the accuracy of the numerical schemes in ASAD. **ptol** is the relative accuracy of tolerance that's used in the time integration schemes (those have that an interactive procedure), whilst **ftol** is the relative accuracy for the family member iteration. By relative accuracy we mean that the solution will be solved until the change between successive iterations is less than **ftol.y**.

Perhaps the only other parameters that the user might want to change are those that fix the name of the chch.d and rat?.d files. All these file names are stored as character parameters in the paramc.h file. Where changing these can come in useful is if you are developing a chemistry scheme and are using a chch.d file in which no families are used and, say, the SVODE integrator (see section 4.7) as a control run, whilst testing a family scenario with the same species using the computationally cheaper IMPACT integrator. This would need two chch.d files; say chch.stiff.d and chch.family.d in which the species types (see section 4.2 and section 7.1) are set differently. You would also need two paramc.h files since the number of advected tracers is different in the two cases; say paramc.stiff.h and paramc.family.h. In each of these files, you can then alter the parameter CHCH to use the correct chch.*.d file. As part of the run script for your model, you would then have to copy the appropriate paramc.*.h file to paramc.h and recompile the model[1]

4.3.1 The PEPS variable

At several places in the ASAD code a variables called **peps** is used to guard against zero divides. We have to compute this, rather than set it as a PARAMETER, to allow for all possible computer hardware and precisions that ASAD might be run at. The LAPACK routine slamch.f (from NETLIB) returns the safe minimum value such that 1/sfmin would not overflow. Since peps is used in ftoy in calculations like y1/y2 where y2 is small we have to ensure that the result never overflows. Therefore, we use the maximum likely value of y1 to scale sfmin returned by slamch. Typically, the maximum total no. density is about $1x10^{19}$ in the troposphere (see routine cinit.f for more details).

1. **IMPORTANT!** As paramc.h controls all the array sizes in ASAD, it's vital that you recompile ALL the ASAD code when paramc.h is changed. One of the most common problems users have with ASAD is having a Makefile which does not recompile all the ASAD code whenever paramc.h is changed.

4.4 Call to subroutine CINIT

Subroutine *cinit* initialises all the variables used in the chemistry. This routine should be called once at the start of a model run. There are four arguments to cinit; an array each for integer, real, logical and character variables. Entries in this array correspond to ASAD variables or switches. This was done so that extra options could be added without altering the argument list for cinit. The following variables and switches should be passed as arguments:

Integer block (array kargs)

kargs(1) = kcdt Chemistry sub-timestep (in seconds). There must be a whole number multiple of chemistry timesteps in one model timestep, otherwise the chemistry timestep is adjusted in order to achieve this requirement.

kargs(2) = kmethod Flag to set the chemistry integration method. Non-stiff methods take values 1-9. Stiff methods take values 10-19. Current options are: (1) IMPACT scheme, (2) Quasi-steady state scheme, (10) NAG BDF stiff integrator, (11) SVODE stiff integrator from NETLIB. With any other value, kmethod is set to zero and only the instantaneous values of the chemistry tendencies are returned. That is, no integration of the chemistry is done. It is possible to add further options for this flag by performing a few simple edits to ASAD (see section 6 and 7).

kargs(3) = kfphot The frequency at which the subroutine photol is called to compute the photolysis rates. Called once if set to zero, otherwise it's called every kfphot chemical substeps. If kfphot is negative, the routine photol is called every -kfphot seconds.

kargs(4) = nit0 The maximum number of iterations in the subroutine FTOY if the chemistry is not integrated separately (see section 7).

kargs(5) = nitfg The maximum number of iterations in the subroutine ftoy when called during the "first guess" stage of the IMPACT integration scheme (see section 7).

kargs(6) = nitnr The maximum number of iterations in the subroutine ftoy when called during the "Newton-Rhapson" stage of the IMPACT integration scheme (see section 7).

kargs(7) = nrsteps The maximum number of Newton-Rhapson iterations in the IMPACT integration scheme (see section 7).

Logical block (array oargs)

oargs(1) = lhet Set .true. if heterogeneous reactions are to be included in ASAD. If this is the case the user must supply subroutine *hetero* (see section 5.4).

oargs(2) = lvmr Set .true. if the calling model uses volume mixing ratios. These will be converted to number densities for the duration of the call to ASAD and converted back before exit.

oargs(11) = ldepd ldepd(jpspec) is a logical array. The first element is taken from oargs(11). Set element i .true. if dry deposition for species i is

on. If this is the case the user must supply subroutines *inddep, drydep* (see section 5.3).

oargs(11+jpspec) = ldepw ldepw(jpspec) is a logical array. The first element is taken from oargs(11+jpspec). Set .true. if wet deposition is to be included in ASAD. If this is the case the user must supply subroutines *inwdep,wetdep* (see section 5.3).

oargs(11+2*jpspec) = lemit lemit(jpspec) is a logical array. The first element is taken from oargs(11+2*jpspec). Set .true. if emissions are to be included in ASAD. If this is the case, the user must supply subroutines *inemit* and *emissn* (see section 5.2) otherwise they are never called.

Real block (array rargs)

rargs(1) = dtime Timestep of calling model (in seconds).

Character block (array cargs)

Unused at present. Reserved for future use.

4.5 Call to subroutine CDRIVE

The subroutine CDRIVE is the driver routine for the chemistry and controls the chemistry integration. It should be called every model timestep. CDRIVE must be called from within a spatial loop if the atmospheric model is either a 2-D or a 3-D model. The following arguments must be passed to cdrive:

cdot Real, dimensioned as (**jpnl, jptrc**). Unused on entry. On exit, contains the tendencies of the families due to chemistry. If the chemistry has been integrated, the tendency is an average over the model timestep (calculated from **cdot = (ftr(out) - ftr(in)) / dtime** where **dtime** is the model timestep, passed in the argument list to subroutine *cinit*). Otherwise, the tendencies are instantaneous values.

ftr Real, dimensioned as (**jpnl, jptrc**). Concentrations in either volume mixing ratio or number density (molecules cm^{-3}) of the model families/tracers at the current timestep. If the chemistry has been integrated, the concentrations are updated to their values at the end of the model timestep.

p Real, dimensioned as **jpnl**. Pressure at centre of gridbox (Nm^{-2})

t Real, dimensioned as **jpnl**. Temperature at centre of gridbox (K).

4.6 Chemical families in ASAD

One of the more unique features in ASAD is its ability to automatically partition chemical families into its constituent members. We assume that the reader is already familiar with the concept of chemical families. If not Austin (1991) has a good description of their use.

Typically, a chemical family in one in which a group of short lived species are parametrized and transported in terms of a longer lived species. For example, we might have an odd oxygen family, O_x, which contains the species: O_3, $O(^1D)$ and $O(^3P)$ where O_3 is the long lived species and $O(^1D)$ and $O(^3P)$ are short lived and assumed to be in steady state.

The O_x family is therefore just the sum of its members

$$O_x = O_3 + O(^1D) + O(^1P) \quad \text{(EQ 8)}$$

It is important to understand that *only* the family concentration is passed to ASAD from the calling model. ASAD will split the family into the member species before using them to calculate the rates of change. The three species, O_3, $O(^1D)$ and $O(^3P)$ are *only* known inside the ASAD code in the array called *y.* They are not transported in the calling model. If you want to access them in the model, include the COMMON block CMSPEC in your code but remember if you tell ASAD your model uses volume mixing ratio (LVMR is .TRUE.) then you must divide the species in *y* by the total number density (array TND in COMMON CMPHYS) as ASAD keeps the individual species in number density. Also, depending on the time integration scheme, the species may not be at the same time level as the tracer/families, stored in the array *f* since *f* is the tracers which are actually integrated in time. The species are only used to compute the tendencies.

The family can be separated into its members as we assume the short lived species are in steady state. That is, if we write the tendency for each species as

$$\frac{dy}{dt} = P + Ly \quad \text{(EQ 9)}$$

where *P* is the sum of the production terms for species *y* and L is the sum of the loss terms then for a steady state species its concentration is simply

$$y = -\frac{P}{L} \quad \text{(EQ 10)}$$

If we denote the major family member (O_3 in the above example) as *Y* then the concentration of the minor family member is given by

$$R_y = \frac{y}{Y} = -\frac{P}{LY} \quad \text{(EQ 11)}$$

We can calculate both the product and loss terms but *Y* the major family member is unknown. We can calculate this from Eq. 8 by dividing by the major family member e.g.

$$\frac{O_x}{O_3} = 1 + \frac{O(^1D)}{O_3} + \frac{O(^3P)}{O_3} \quad \text{(EQ 12)}$$

which if we invert gives

$$O_3 = \frac{O_x}{1 + R_{O1D} + R_{O3P}} \quad \text{(EQ 13)}$$

The family members are computed in the subroutine FTOY (families to species). Solving for them involves an iterative process. Initially we partition the family species such that the major members is equal to the tracer value and the minor species are zero (e.g. we set $O_3 = O_x$ and $O(^1D)=O(^3P)=0$. This allows us to compute the ratios, *R*, to give a better guess for the major family species from which we can calculate the minor species concentrations. All of this iterative procedure is contained inside the FTOY subroutine.

4.6.1 Calculating the family ratios

A word about the way ASAD computes the ratios is probably warranted. Traditionally in atmospheric chemistry programs, in computing the production (*P*) and loss (*L*) terms for each species, *only* the interchange reactions are considered (i.e. those that produce no net loss from the family) and other reactions known to be important in loss cycles. This considerably reduces the amount of computation needed. Furthermore, for some families, it is possible to compute the ratios analytically through carefully ordering the calculation, rather than use an iterative procedure.

However, in ASAD we use *all* of the reactions for each family member in calculating the ratio. It is impossible for ASAD to know which reactions could be excluded and which are important to include if they play a role in important catalytic loss cycles. From a programming point of view it was also easier to write the code to use all of the reactions. Thus, in ASAD, the *true* steady state value of the minor, steady state species is calculated. We would not expect this to give results very different from those obtained using just the family interchange reactions, particularly for the major family species (because the concentration of such species is so large compared to the minor family species - and if it did make a difference the family approach would break down). Unfortunately, this has the impact that the code in **ftoy** can be computationally expensive.

The iteration to compute the ratios and steady state species is controlled by two parameters (see paramc.h). Convergence is achieved when the difference between two iterations gives a value less than RTOL*x*y or when y is less than ATOL, where RTOL is the relative tolerance and ATOL is the absolute tolerance. The user is free to choose the values of RTOL and ATOL but the ones set by default are known to give good results.

Since the values of the species, y, are bounded by the use of ratios, the family members' concentrations converge within a small number of iterations. However, during this iteration, the concentrations of the steady state species which are not a member of a family must also be computed. The concentrations of these species are usually small and often strongly dependent on each other as well as the family members. Consequently, it is generally the convergence of the steady-state species which determines the number of iterations required. The user must experiment to determine the optimum number required for the desired accuracy.

ASAD also has the facility to treat species as tracers when their chemical lifetimes are long enough, but as members of a family when their lifetime becomes short compared to the timestep. An example is the species NO_3 in the lower troposphere. By day, its lifetime is of the order of seconds but by night it is roughly 12 hours or more. ASAD can include the species in the family when the lifetime is short and integrate it as a separate tracer when its lifetime increases. Note that in this case, the calling model is transporting this species as a tracer, even though at times it is a family member. However, the use of this option can introduce a computational overhead. This is because, for some of the time integration subroutines, it is possible to precompute coefficients once only at the start of the run. If a species moves in and out of a family during the run, these coefficients must be recomputed at each timestep.

4.7 Time integration schemes

ASAD has a choice of built-in integrators which can be used to solve the ODEs resulting from Eq. 3. Integrators designed for stiff and non-stiff problems are available. The choice of method to use will depend on the required accuracy of the solution and the computational cost of the chemistry. For instance, if families are not used, the problem is generally stiff. A chemistry based on families used in a 3D atmospheric transport model could use one of the computationally efficient non-stiff solvers.

The first non-stiff integrator is the Quasi-Steady State Approximation (QSSA) as described in Hesstwedt et al (1978). This is a cheap and explicit time integration method which is well known in the literature. It is included in ASAD as it is a popular method. However, it can suffer from poor conservation and we do not recommend its regular use without careful testing and possibly modifications to the source code to improve the conservation of the scheme.

We have included the IMPACT algorithm as described in Carver and Stott (1997). This is an accurate implicit time integration method with good conservation properties, whilst efficient and highly suited to modelling stratospheric and tropospheric chemistry in 2D and 3D models when families are used to reduce the overall stiffness of the chemical system.

Two highly accurate but expensive integrators suitable for stiff problems may also be chosen by the user. The first is the NAG library routine D02EAF (NAG 1996) and the second is the SVODE integrator (Brown et al. 1989) from the NETLIB subroutine library. The NAG integrator can only be used if the NAG library is available; it is not provided as part of ASAD. The call to this routine is commented out by default to avoid compilation problems. The SVODE subroutine is included with ASAD. The SVODE integrator can solve both stiff problems, using backward differentiation formulae, or non-stiff problems using an Adams method. Both of these options are provided in ASAD.

It is very important to note that if either the SVODE or NAG integrators are selected, they will only solve the equations for a gridpoint at the time. Since, the inner loop is over gridpoints in the ASAD code, this means that the ASAD subroutines which

represent most of the computational work will not vectorize and the performance will suffer on a vector machine. We therefore do not recommend the use of these integrators for multi-dimensional problems on vector processor architectures.

A typical scenario for the development of an atmospheric chemistry scheme would be to use either the SVODE or NAG integrator initially for testing and revising the chemical species and reactions in a box model of the atmosphere. It could also be used to compare the results when chemical families are used. Then, the computationally cheaper integrators could be used after development when the chemistry scheme is used in a 2D or 3D model.

5 User supplied subroutines

It is necessary for the user to supply subroutines to calculate photolysis rates, and if desired, emissions, deposition, heterogeneous reaction rates and a subroutine to set the constant field values (type ‘CF’ in the chch.d file). In this section, we outline the form that these subroutines must take and suggest ASAD common blocks that may be useful. Clearly, it would be sensible for the user to familiarise her/himself with the ASAD code before embarking on this task. In case of any problems or questions please refer back to the maintainer of the ASAD package, via the WWW pages (see introduction).

The following common blocks may be useful for all the user supplied routines:

paramc	Contains parameters used to dimension arrays, including **jpnl** and **jpspec** (see section 4.3).
cmphys	Contains the physical quantities: pressure, temperature, total number density.

5.1 The rate array, rk.

Although ASAD uses separate files for the different types of reactions; bimolecular, termolecular and so on, internally a single array, called rk is used to hold the reaction rates. Also, when the reactions are read in, they are reordered somewhat so that the user cannot assume that the first bimolecular reaction is stored in index 1, the 2nd reaction in index 2 of the rk array etc. The reordering and use of a single array is done to make the code more efficient.

Unfortunately, this does introduce an extra level of complexity to the user supplied code. Indexing arrays are provided so that assigning reaction rates to the rk array can be done easily, but indexing the rk array has be to done indirectly. What does this mean in practise? As an example, suppose you wanted to assign the reaction rate for the first photolysis reaction. You might try something like:

```
do j = 1, jpnl
rk(j,1) = 3.503E-11
end do
```

However, you must never assume that the reactions as listed in the rates files are stored in the rk array in the same order. Our example above should read:

```
do j = 1, jpnl
rk(j,nprkx(1)) = 3.503E-11
end do
```

The array nprkx() is used to get the correct index for the first photolysis reaction. For example, if nprkx(1) = 3, then the first photolysis reaction rate is actually stored in rk(j,3).

As far as the user is concerned, you only need to worry about this for the photol and hetero subroutines. The bimolecular and termolecular rates are already done this way. For the heterogeneous rates, use the indexing array nhrkx. See cominx.h for more details.

5.2 Subroutines inphot & photol

Since photolysis will always be included in an atmospheric model, these routines are always called. ASAD contains dummy routines that must be removed or overwritten. Subroutine *inphot* is called once at the start of a model run from *cinit*. This subroutine is intended to initialise the variables used in the user supplied photolysis scheme. Not all photolysis schemes will need such an initialisation routine, in those cases the dummy routine can simply be left untouched.

Subroutine *photol* is the interface routine between ASAD and the user supplied photolysis scheme. Photol is called from *cdrive*, at a frequency determined by the input variable nfphot. It should be used to set the photolysis rates, corresponding to the reactions in the photolysis ratefile, *ratj.d*. The common block **cmreac** may be useful because that contains the data read from the photolysis ratefile.

5.3 Subroutines inemit & emissn

These subroutines need only be supplied if emissions are to be included with the chemistry, and are only called if any elements in the logical switch array *oemit* passed to the subroutine CINIT are set to .true..

Subroutine INEMIT is called once at the start of a model run from CINIT. It should be used to initialise all variables relating to emissions and to read the necessary files and data into working arrays.

Subroutine EMISSN is called once each model timestep from CDRIVE. It should be used to calculate the emission rates of the chemical species, **emr** (molecules $cm^{-3}s^{-1}$). The array **emr** is dimensioned as (**jpnl, jpspec**) and is passed via the common block **cmdpem**.

For more details see section 7.11.

5.4 Subroutines inddep, inwdep, drydep & wetdep

These subroutines need only be supplied if deposition to the earth's surface is to be included with the chemistry, and are only called if any elements in the logical switch arrays **ldepd** or **ldepw** passed to the subroutine CINIT are set to .true..

Subroutines INDEPN and INWDEP are called once at the start of a model run from CINIT. They should be used to initialise all variables relating to dry and wet depositions respectively, and to read the necessary files and data into working arrays.

Subroutines DRYDEP and WETDEP are called once each model timestep from CDRIVE. They should be used to calculate the dry and wet deposition rates of the chemical species, **dpd** and **dpw**(s^{-1}). These arrays are dimensioned as (**jpnl, jpspec**) and are passed in the common block **cmdpem**.

5.5 Subroutine hetero

This subroutine need only be supplied if heterogeneous reactions are to be included in the chemistry, and is only called if the logical switch **lhet** passed to the subroutine CINIT is set to .true..

Subroutine HETERO is called every chemistry timestep from the time integrator routine (or from CDRIVE if the chemistry is not integrated). It is called every chemistry timestep in order to accommodate those heterogeneous reaction schemes that depend on tracer concentrations. HETERO should be used to calculate the heterogeneous rate coefficients (cm^3s^{-1}) for those reactions in the heterogeneous ratefile, *rath.d*. The common block **cmreac** may also be useful because that contains the data read from the heterogeneous ratefile.

See section 7.10 for more details.

5.6 Adding constant field types (‘CF’)

There are two ways to handle species which are constants in ASAD. The relevant types for the chch.d file are ‘CT’ and ‘CF’. Any species of type CT is set to a constant and this applies at every gridpoint. However, the CF field, which stands for ‘constant field’ allows for a different value at every gridpoint. For example, you might want to set H_2O (water vapour) to some analysed gridpoint field. The type CF allows you to do this.

If any of your species is of type ‘CF’, then ASAD will call a subroutine called INICNT (initialise constant). This subroutine is passed 4 arguments:

```
subroutine inicnt(species, y, klen, kcdt)
```

where species is the name of the species to be initialised, as listed in the chch.d file, y is the array to be initialised, klen is the number of points to initialise (in other words, the size of the array y) and kcdt is the current chemical (not the dynamical) timestep. If more than one species is of type CF, ASAD will call INICNT once for each species. INICNT is called at the start of every chemical timestep. It is up to you, the user to place the appropriate code in your version of INICNT to deal with multiple species of type CF i.e. you’ll need some sort of IF-THEN-ELSE logic to test the *species* argument before assigning the array.

The dummy subroutine INICNT included with the ASAD code includes an example of how this subroutine might be written. Note two things however. First, don’t include any I/O in this subroutine (except possibly on the very first call) as ASAD repeatedly calls this during the timestepping and I/O would probably have a significant impact on the runtime of the code. Second, don’t be tempted to modify the ASAD call to INICNT to add extra arguments for instance. This would mean making changes to

several ASAD subroutines, which makes upgrading to new versions of ASAD more difficult. It is better to use COMMON blocks to pass any variables or arrays you might between the calling model and the INICNT subroutine. Again, see the example given in the dummy routine INICNT supplied with ASAD.

6 Editing ASAD

Generally, it should not be necessary to edit ASAD. In most cases, the package can be used as a black box. However, advanced users may wish to change certain aspects, add the option of different integration schemes for instance or simply change the fixed concentrations of species like CO_2. We have endeavoured to structure the code in a thoughtful and logical sense. In most cases, any changes will simply require the addition of a few extra lines of code. However, we strongly advise users to familiarise themselves with the ASAD code before making any changes.

Users are welcome to contact the maintainer of the ASAD code in case of questions. You can send email using the World-Wide Web ASAD pages on: http://www.atm.ch.cam.ac.uk/acmsu/asad/.

There have been cases when people have altered the ASAD code to add more arguments to subroutines like emissn or drydep for their emission or deposition schemes. We strongly discourage this because it not only runs the risk of introducing bugs but it also makes upgrading to future versions of ASAD more tricky. Instead we advocate the use of COMMON blocks to pass any needs variables between user supplied code to ASAD and the atmospheric model. An example of where this might be needed is in setting field values for constant field type species (type ‘CF’). See section 5.6.

The only piece of code the user is expected to change is the parameters in paramc.h. It is extremely important to correctly set the number of species, tracers and the number of reactions in each of the ratefiles. The file paramc.h distinguishes between parameters that the user is free to change and those that the user would not normally need to alter. For example, parameters which control the size of some of the internal arrays are usually not altered by the user. However, if some internal arrays are too small, ASAD will catch this, stop and print a message detailing the parameter that needs to be increased. There are several other parameters that the user may alter if they desire. These are discussed in more detail in section 4.3.

7 Setting up an atmospheric chemistry model: A step-by-step example

As an example, we now describe step-by-step how we set up a tropospheric chemistry scheme based around the oxidation of CH_4 using SELECT and the ASAD reaction database for use in an atmospheric model.

7.1 Stage 1. Selecting a chemistry.

The first stage in setting up an atmospheric chemistry scheme is to determine which species to include in the reaction network. We defined the *chch.d* file shown below in Table 4.

We will discuss the choice of option flags later since these are unimportant when the file is used in SELECT. The names for the species were checked against those in the master list of species, *species.d*, to ensure that the correct form was used. For example, CH_3OOH is written as MeOOH in the ASAD database. We were careful to select all the species that we anticipated would be involved in the chemistry. We also placed the major species of a family after the minor members, including those species that switch in and out of a family depending on their lifetime - this is important for ASAD.

Table 4: An example chch.d file for methane oxidation

1	‘O(3P)’	1	‘FM’	‘Ox’
2	‘O(1D)’	1	‘FM’	‘Ox’
3	‘O3’	1	‘FM’	‘Ox’
4	‘N’	1	‘FM’	‘NOx’
5	‘NO’	1	‘FM’	‘NOx’
6	‘NO3’	1	‘FM’	‘NOx’
7	‘N2O5’	2	‘FT’	‘NOx’
8	‘HO2NO2’	1	‘FT’	‘NOx’
9	‘NO2’	1	‘FM’	‘NOx’
10	‘HONO2’	1	‘TR’	‘ ‘
11	‘H’	1	‘SS’	‘ ‘
12	‘OH’	1	‘SS’	‘ ‘
13	‘HO2’	1	‘SS’	‘ ‘
14	‘H2O2’	1	‘TR’	‘ ‘
15	‘CH3’	1	‘TR’	‘ ‘
16	‘CH4’	1	‘TR’	‘ ‘
17	‘CO’	1	‘TR’	‘ ‘
18	‘CO2’	1	‘CT’	‘ ‘
19	‘HCO’	1	‘SS’	‘ ‘
20	‘HCHO’	1	‘TR’	‘ ‘

Table 4: An example chch.d file for methane oxidation

21	‘MeO’	1	‘SS’	‘ ‘
22	‘MeOO’	1	‘SS’	‘ ‘
23	‘MeOOH’	1	‘TR’	‘ ‘
24	‘H2O’	1	‘TR’	‘ ‘
25	‘O2’	1	‘CT’	‘ ‘
26	‘N2’	1	‘CT’	‘ ‘

7.2 Stage 2. Running SELECT

To run SELECT, we edit the default job script *sel_job* to specify that 26 species were included in the *chch.d* file. There is no need for us to change the names of the master ratefiles because we were using those from the ASAD reaction database. Later we will use the results from this run of SELECT as our own ‘master’ ratefiles. We take copies of the master ratefiles so that they appear in the same directory as the job script. We decide to set all three logical flags in the job script to .true. so that we obtain ratefiles for both “open” and “closed” reaction networks, and a file listing the difference between the two. We are not including heterogeneous reactions in our scheme and so we removed that section from the job script. The script we used is shown below:

```
#! /bin/sh
#-------------------------------------------
f77 -o selx select.f
/bin/rm control
#-------------------------------------------
# JOB SCRIPT FOR RUNNING *SELECT*
# Paul D.Brown, CAS, Cambridge. 18/2/94
# For each section below specify:
#--
# filename of master ratefile
# number of species in chch.d
# 3 logical switches:
# open, closed, difference
#--
# The user must supply chch.d - a list of
# species in ACMSU format.
# The output files will be for an open
# and/or a closed reaction network
# and/or the difference.
# The user should read info.doc
# after running this job.
#-------------------------------------------
echo bimolecular reactions...
cat > control << EOF
bimol.d
26
.true. .true. .true.
EOF
selx < control
#-------------------------------------------
echo trimolecular reactions...
cat > control << EOF
trimol.d
26
.true. .true. .true.
EOF
selx < control
```

```
#-------------------------------------------
echo photolysis reactions...
cat > control << EOF
photol.d
26
.true. .true. .true.
EOF
selx < control
#-------------------------------------------
 /bin/rm control selx
echo
echo Read the information file: info.doc
#-------------------------------------------
```

The script was run using the UNIX command:

% sel_job

Note to Apple MacIntosh users: Apple MPW (MacIntosh Programming Workshop) should support the use of Unix style redirection (‘<‘) as used in this job script so that the fortran program reads the input on channel 5. If you cannot get this to work you will need to edit the select.f code to change the definition of channel 5 (e.g. open(5,file=’select.input)).

Running this job generates the following output files:

ratb.open, ratb.clos, ratb.diff, ratt.open, ratt.clos, ratt.diff ratj.open, ratj.clos, ratj.diff, info.doc

It is sensible to check the *info.doc* file because this gives information on the chosen chemistry - so long as the option flags have been set correctly. The contents of our info.doc file are listed below:

```
 ** SELECT - INFORMATION FILE **
For your chosen species set, the following advected tracers have been identified:

 1: Ox          2: NOx         3: N2O5        4: HO2NO2      5: HONO2
6: H2O2         7: CH3         8: CH4         9: CO          10: HCHO
11: MeOOH       12: H2O

Tracers *MUST* be initialised in the model in the above order.
The major member of each of the families is given below. If this is not acceptable,
then you must reorder the species in chch.d so that the major species follows the others.

Ox (O3)
NOx (NO2)
```

The tracers in the atmospheric model **must** be initialised in the order given by the *info.doc* file, subject to any subsequent editing or reordering of the *chch.d* file. We have taken care to place the major species of a family after the minor members, and we receive confirmation in *info.doc* that ASAD would treat O_3 and NO_2 as the major species in the O_x and NO_x families respectively.

7.3 Stage 3. Editing the chch.d file.

This stage is not always necessary. If the correct option flags have already been set in the *chch.d* file, and if the user wants to include all the listed species, then the file can be used in its original form. However, in this example we decide to remove some of the species from the chemistry. This has the benefit of reducing the number of reactions slightly and, possibly, the number of species. Thus, we may hope to save cputime when running the model. The only species that can be removed are those that react immediately on formation, usually with O_2 or N_2, or that rapidly decompose thermally. The species we choose to remove are H, N, CH_3 and HCO, which react rapidly with O_2 to form HO_2, NO+O(^{3}P), MeOO and CO+HO_2 respectively.

To remove species, it is necessary to do the following:

1. Remove the species from the *chch.d* file, renumbering the remaining species afterwards. Note that the species can be in any order in the chch.d file, as long as the major family member is always listed last for any particular family.
2. Remove all reactions from the ratefiles which SELECT output, in which the “removed species” is one of the reactants.
3. Edit the ratefiles for all occurrences of the “removed species” as products of a reaction. For example, in all reactions in which HCO was formed, we have to change the string ‘HCO’ to ‘CO HO2’, taking into account the format of the files. See section 3.4 for more details.
4. Renumber the edited ratefiles using the RENUMB program.

We would have liked to remove the species MeO in addition to those above. However, this would have meant including a bimolecular reaction with four products. The format of the ratefiles is controlled by parameters in paramc.h. In principle we could have increased the relevant parameter and rebuilt our ratefiles so that they all had 4 products. However, at the time of writing, we’ve never tested this so we chickened out! Our revised *chch.d* file is shown below:

Table 5: Revised example chch.d file

1	‘O(3P)’	1	‘FM’	‘Ox’
2	‘O(1D)’	1	‘FM’	‘Ox’
3	‘O3’	1	‘FM’	‘Ox’
4	‘NO’	1	‘FM’	‘NOx’
5	‘NO3’	1	‘FM’	‘NOx’
6	‘N2O5’	2	‘FT’	‘NOx’
7	‘HO2NO2’	1	‘FT’	‘NOx’
8	‘NO2’	1	‘FM’	‘NOx’
9	‘HONO2’	1	‘TR’	‘ ‘
10	‘OH’	1	‘SS’	‘ ‘
11	‘HO2’	1	‘SS’	‘ ‘

Table 5: Revised example chch.d file

12	‘H2O2’	1	‘TR’	‘ ‘
13	‘CH4’	1	‘TR’	‘ ‘
14	‘CO’	1	‘TR’	‘ ‘
15	‘CO2’	1	‘CT’	‘ ‘
16	‘HCHO’	1	‘TR’	‘ ‘
17	‘MeO’	1	‘SS’	‘ ‘
18	‘MeOO’	1	‘SS’	‘ ‘
19	‘MeOOH’	1	‘TR’	‘ ‘
20	‘H2O’	1	‘TR’	‘ ‘
21	‘O2’	1	‘CT’	‘ ‘
22	‘N2’	1	‘CT’	‘ ‘

The choice of flags in the *chch.d* file is now of critical importance. These flags determine how ASAD treats the various species. We chose to have two families, O_x and NO_x. From previous experience, we anticipated that near the earth’s surface the thermal decomposition rate of HO_2NO_2 and N_2O_5 may be sufficiently fast (and their lifetimes short) that integrating these species separately may lead to stability problems. However, both species have long lifetimes in the upper troposphere and stratosphere, too long for them to be included in a family. Therefore we chose to define both these species as type ‘FT’ which means ASAD will place them in the NO_x family if their lifetime in any model timestep becomes shorter than half the chemistry timestep. We chose to have a number of species in steady state. This saved us from keeping their concentrations in the model, or looked at another way, reduced the number of advected tracers. The concentrations of these species are low enough that neglecting their transport will not introduce any great inaccuracies. We set the concentrations of O_2, N_2 and CO_2 to constants. If we had wanted to set species other than O_2, N_2, H_2 and CO_2 to constants, or if we had wanted to change the values of the constants used by ASAD, we would have been forced to edit the ASAD subroutines CINIT and FYINIT, and the common block **cmspec**.

Editing the *chch.d* file may change the number of “model tracers” listed in the *info.doc* file generated by SELECT. The user must be aware of this when initialising the tracers in the atmospheric model. Our edits had the effect of removing CH_3 from the list given in our *info.doc*.

7.4 Stage 4. Editing the ratefiles.

We have already done some editing of the ratefiles in removing the species as discussed above.

The photolysis ratefiles generated by SELECT from the ASAD reaction database, *ratj.open* and *ratj.clos*, were both unsatisfactory. Many of the photolysis reactions in the database have unknown products. We decided to design our own photolysis ratefile, *ratj.d*, for use with our example scheme. Any new ratefiles must always be in

the same format as their equivalents in the ASAD database. To help reformat the ratefiles, we refer to the format statements used in the ASAD subroutine READRAT. These are: for bimolecular, photolysis and heterogeneous ratefiles:

```
920   format(1x, i4, 1x, 4(a10, 1x), a10, 1pe8.2, 1x, 0pf5.2, 1x, f8.1)
```

for termolecular ratefiles:

```
930   format(1x, i4, 1x, 4(a10, 1x), f7.2, 1x,
     *       2(1pe8.2, 1x, 0pf5.2, 1x, 0pf8.1, 1x))
```

Our photolysis ratefile, *ratj.d*, is shown below:

Table 6: Photolysis ratefile

1	O2	PHOTON	O(3P)	O(3P)		0.00E+00	0.00	0.0	J2
2	O3	PHOTON	O2	O(3P)		0.00E+00	0.00	0.0	J3
3	O3	PHOTON	O2	O(1D)		0.00E+00	0.00	0.0	J3A
4	H2O2	PHOTON	OH	OH		0.00E+00	0.00	0.0	JH2O2
5	HONO2	PHOTON	OH	NO2		0.00E+00	0.00	0.0	JHNO3
6	HO2NO2	PHOTON	HO2	NO2		0.00E+00	0.00	0.0	JPNA
7	NO2	PHOTON	NO	O(3P)		0.00E+00	0.00	0.0	JNO2
8	NO3	PHOTON	NO	O2		0.00E+00	0.00	0.0	JNO3A
9	NO3	PHOTON	NO2	O(3P)		0.00E+00	0.00	0.0	JNO3B
10	N2O5	PHOTON	NO3	NO2		0.00E+00	0.00	0.0	JN2O5
11	HCHO	PHOTON	HO2	CO	HO2	0.00E+00	0.00	0.0	JC2OA
12	HCHO	PHOTON	H2	CO		0.00E+00	0.00	0.0	JC2OB
13	MeOOH	PHOTON	MeO	OH		0.00E+00	0.00	0.0	JMHP

The products of reaction 11 should actually be H + HCO. However, since we have removed both these species from our *chch.d* file, we had to substitute the products HO_2 (for H) and CO + HO_2 (for HCO). The alert reader will notice that the ratefile is not closed with respect to our chosen species set. H_2 is not one of the species listed in our *chch.d* file, but is formed in our photolysis reaction number 12. We were not concerned about the production of this species but we did need to include the photolysis of HCHO. This is a good example of when an “open” reaction network is preferable to a “closed” one. We added a character string at the end of each line for use in our photolysis scheme, in place of the character string used in the ASAD ratefiles as references or notes. ASAD reads the string but does not store, so in order to use the string, we will have to reread the file in the user defined subroutine INPHOT by using the ASAD subroutine READRAT (see the routine INRATS for examples of its use).

After scanning the *ratb.diff* and *ratt.diff* files for the difference between the open and closed bimolecular and trimolecular reactions, we decided to use the open ratefiles as the basis for our ASAD ratefiles. This is because they included reactions that we

wished to retain, even though not all the products appeared in our *chch.d* file. For example, in the termolecular reactions we wanted to include this important loss of $O(^1D)$, even though we were not concerned with N_2O:

$$O(^1D) + N_2 \rightarrow N_2O \qquad \text{(EQ 14)}$$

In order to keep a record of the various stages in constructing the chemistry scheme, we copied the open ratefiles before any editing. We then did the following:

1. Deleted or edited reactions to account for removing certain species (see above)
2. Deleted reactions with upper limits to their rates (symbolised by 'U' in the comments column), so that they could not influence the chemistry.
3. Deleted most reactions with unknown products, so as to maintain conservation. However, in one case we made an educated guess as to the products because the reaction represented an important loss route for one of the reactants.
4. Deleted most of the reactions in which the products were not included in the *chch.d* file, *not all* otherwise we may as well have used the "closed" set of reactions. In one case, this involved deleting two branches out of three. In order to maintain the true loss rate for the reactants, the rate for the remaining branch was increased to equal the sum of the three branches. This meant overestimating the rate of formation of the products from this branch, but we felt that this was acceptable in this case.

These edits reduced the number of bimolecular and trimolecular reactions from 65 and 19 in the original *rat?.open* files to 40 and 12 respectively. We then renumbered the reactions in the ratefiles by compiling and running the program RENUMB:

```
% f77 -o renumb renumbr.f
% renumb
RENUMBER reactions; ensure line following reaction
data is either blank or is a null reaction with index = 9999
input ratefile? (<10 characters)
> ratb.dat
output ratefile? (<10 characters)
> ratb.d
Is the ratefile in trimolecular format? (y/n)
> n
```

We now had a *chch.d* file and three ratefiles, *ratb.d, ratt.d* and *ratj.d,* ready for use with ASAD.

7.5 Stage 5. Setting up the atmospheric model.

Having put the ratefiles and *chch.d* file in their final form for use with ASAD, it is necessary to set up the atmospheric model for use with ASAD. ASAD uses arrays with one spatial dimension to allow for vectorisation. In a box or trajectory model, this spatial dependency can be ignored simply by setting the index to 1 throughout. In a column model, 2-D or 3-D model, the user has to decide whether to pass arrays over longitude, latitude, altitude or a combination of these. The advantage of passing arrays over level is that the zenith angle will be the same for all points, and the iterated

concentrations in FTOY may converge at all points at around the same time saving cputime. On the other hand, using longitude or latitude will generally involve longer arrays and may lead to increased efficiency in the vectorisation. In our step-by-step example, let us assume that we are using a 3-D model and that we wish to vectorise over longitude.

Although we had a good idea which tracers/families we would be carrying in our model at the time we defined our *chch.d* file, the list was confirmed in the *info.doc* file produced by SELECT. Furthermore, this file gave information on the order in which ASAD would expect those tracers/families to be listed. We must take care to keep the model tracers/families in that order - for example, when we initialise them. Bear in mind, the list may change following editing of the *chch.d* file. It is always possible to rerun SELECT to obtain a new *info.doc*, but ignore the ratefiles generated in this second run and ensure that the originals are not overwritten. Since we removed CH_3, we were left with 11 tracers/families in our model (see Stage 1).

7.6 Stage 6. Setting ASAD parameters.

Moving ever closer. We now have to set the ASAD parameters used to dimension variables and to determine loop lengths. This entails editing the ASAD parameter block **paramc**. We set the following parameters:

jpctr = 11	The number of tracers/families in the model, taken from *info.do* but keeping in mind the edits subsequently made to the *chch.d* file.
j**pspec = 22**	The number of individual species involved in the chemistry, i.e. the number of species in the *chch.d* file.
jpnl = 64	The length of the spatial dimension used in ASAD. In our example, this was the number of longitude points. It could equally well be the number of latitude or level points. In a box/ trajectory model, simply set **jpnl = 1**.
j**pbk = 40**	The number of bimolecular reactions in *ratb.d*.
jptk = 12	The number of trimolecular reactions in *ratt.d*.
jppj = 13	The number of photolysis reactions in *ratj.d*.
jphk = 0	The number of heterogeneous reactions in *rath.d*. In our example, we do not have any heterogeneous reactions, so it will not be necessary for us to supply a ratefile.

We keep all the other parameters at their original settings. Other parameters we may consider changing are **pmin, ptol** and **ftol**. The second of these is the tolerance accepted in the convergence tests in subroutine IMPACT whilst the third is the tolerance for the subroutine FTOY, whereas **pmin** is the lowest volume mixing ratio to consider in those tests. In practise, we have found these rarely need changing. However, it's important to appreciate that **ftol** and **ptol** will affect the run time required by the model. Generally speaking, the more accuracy you ask for the more iterations the schemes will need and therefore the more expensive the code becomes.

We could also change the parameters **chch**, **ratb**, and so on. These control the names of the ratefiles. If we wanted to test both a family and non-family chemistry, we could use a different paramc.h (i.e. paramc.family.h and paramc.nofams.h) for each case. We would only need to create 2 chch.d files (i.e. chch.family.d and chch.nofams.d) which have different switch settings. By altering the name of the file using the parameter in our two paramc.h files, this different file would be picked up automatically.

7.7 Stage 7. Model-ASAD interface: cinit.

It is necessary to call the ASAD subroutine CINIT once only from the start of a model run. This routine is used to read files, initialise variables related to the chemistry, and through calls to other subroutines, emissions, depositions and photolysis. From an initialisation subroutine in our atmospheric model, we include the line:

```
call cinit(kargs,oargs,rargs,cargs)
```

Before calling this subroutine, we must set the elements of these arrays to the values required. See section 4.4 for more details.

We decide to use the IMPACT integration scheme. In order to include another scheme, the user should edit subroutine CDRIVE, so that the schemes' driver routine is called for a different value of **method**. The driver routine would have to be supplied.

Trial and error is the best way to determine the parameters used in the IMPACT integration scheme. Experiments in a box model over a range of conditions will indicate the size of the timestep that will maintain sufficient accuracy, say 15 minutes so **ncdt=15*60**. If not integrating the chemistry, **nit0** is the maximum number of FTOY iterations. Play safe and use **nit0=20** or more. If convergence is achieved sooner, the iterations will end anyway. **nitfg** and **nitnr** are the maximum number of iterations in **ftoy** during the "first guess" and "Newton-Rhapson" stages of the timescheme. They should be chosen to compromise reducing cputime with retaining sufficient accuracy in the results. For a limited chemistry like our example, we may get away with values of **nitfg=5** and **nitnr=3**. The actual choice should be made in conjunction with that of **nrsteps**, the maximum number of Newton-Rhapson iterations in the timescheme, which has similar constraints. We used **nrsteps=5**. In our stratospheric chemistry scheme, involving both bromine and chlorine, we found that we had to use **nrsteps=20**, **nitfg=10** and **nitnr=10** in order to obtain reasonable results, maintain conservation, etc. The point is that the choice of values for these parameters and for the chemistry timestep is very much model and chemistry dependent. The variable **nrsteps** is probably the most important as it is directly related to the integration scheme's conservation properties. The variables **nitfg** and **nitnr** determine the accuracy to which the steady state species and family members are computed.

The model timestep, **dtime**, is passed to the chemistry in order to determine how many chemistry steps to take per model timestep. If the choice of the chemistry timestep, **ncdt**, does not give a whole number of steps, it is adjusted and the new, used value is returned.

7.8 Stage 8. Model-ASAD interface: cdrive.

Subroutine CDRIVE is the main driver routine for the chemistry. It should be called every model timestep. In a 2-D or 3-D model, it is necessary to call it from within (a) spatial loop(s) because, as we have already discussed, ASAD only deals with one spatial dimension at a time. In our 3-D model, vectorized over longitude, for example, we have the following structure:

loop over model timesteps
loop over latitude rows
loop over levels
call cdrive (arrays dimensioned over longitude)
end of level loop
end of latitude loop
end of timestep loop

The call to cdrive takes the form:

```
call cdrive(cdot, ftr, p, t)
```

where **cdot** and **ftr** are both dimensioned as (**jpnl, jpctr**), and **p, t** are dimensioned as (**jpnl**). The tendencies of the model tracers/families due to chemistry, emissions and deposition are returned via **cdot**. If the chemistry is not integrated (determined by value of **method** in call to CINIT), these are instantaneous value and the model's integration scheme must be used with a sufficiently short timestep to ensure stability and accuracy in the chemistry component. On the other hand, if the chemistry has been integrated, the array **cdot** holds average values over the model timestep. If these tendencies are added to tracer/family tendencies due to transport mechanisms, before integrating the total tendency, there should not be a problem (though the combination of transport and chemistry may create its own problems). **ftr** contains the tracer/family concentrations. On exit, it is updated to hold the values at the end of the chemistry. These can be used in models where the chemistry and transport are always integrated separately, one after the other. **cdot** and **ftr** can be expressed in terms of either volume mixing ratios or number densities (molecules cm^{-3}), and this should determine the choice of the logical switch **lvmr** passed to CINIT. The variables **p,** and **t** are the pressure (Nm^{-2}) and temperature (K) respectively.

7.9 Stage 9. Supplying a photolysis scheme.

ASAD requires the user to supply photolysis rates corresponding to the reactions in the photolysis ratefile, *ratj.d*. The ASAD dummy subroutine PHOTOL must be replaced by a user supplied subroutine that acts as an interface to the photolysis scheme. The photolysis rates must be written to the array **rk** in units of s^{-1}, as described in section 5.1. A number of other ASAD common blocks contain variables that may be useful in the photolysis scheme. These include **paramc** (parameters), **cmreac** (the ratefile data) and **cmphys** (temperature, pressure and total number density).

ASAD also contains a dummy subroutine INPHOT, which is called at the start of a model run. A user, may if (s)he wishes, replace this dummy routine with one of her/his own in order to initialise the photolysis scheme.

7.10 Stage 10. Supplying a heterogeneous reaction scheme.

If the logical switch **lhet** is set to .true. in the call to CINIT, ASAD calls a subroutine HETERO every chemistry timestep. This routine must be supplied by the user, and in a similar way to the photolysis code discussed above, it must be used to set the heterogeneous reaction rates corresponding to the reactions in the ratefile, *rath.d*. This includes any variation in the rates due to changing circumstances such as the formation of polar stratospheric clouds in a stratospheric chemistry scheme, or aqueous phase chemistry in a tropospheric chemistry scheme. The rates, in units of cm^3s^{-1}, must again be written to the array **rk**, using the index array **nhrkx**. No dummy routines are supplied with ASAD and any initialisation of the scheme must be done from within the subroutine HETERO or within CINIT if any files need to be read.

7.11 Stage 11. Supplying emissions and deposition rates.

If the appropriate logical switches are set .true., ASAD calls the subroutines INEMIT and EMISSN, and INDDEP, INWDEP, DRYDEP and WETDEP. These subroutines must be supplied by the user. Subroutine EMISSN should be used to set the emission rates for each of the chemical species along the spatial dimension used in ASAD, **emr**, in units of molecules $cm^{-3}s^{-1}$. Similarly, the subroutines DRYDEP and WETDEP should be used to set the dry and wet deposition rates, **dpd** and **dpw**, respectively, in units of s^{-1}. All three arrays are dimensioned as (**jpnl, jpspec**) and appear in the common block **cmdpem**. The subroutines INEMIT, INDDEP and INWDEP should be used to initialise the emissions, dry and wet deposition schemes respectively, including reading any datafiles.

Since ASAD has no knowledge of whether the spatial dimension it is using is longitude, latitude or altitude, it may be necessary in a 2-D or 3-D model for the user to pass information on the other spatial dimension(s) to the emissions and deposition schemes via a common block. For example, in our 3-D model vectorized over longitude, we would pass indices for the current level and latitude. Thus, if we only had surface emissions in our model, we would know to set **emr=0** unless the level index indicated that we were dealing with the atmospheric layer immediately above the surface. Suppose those surface emissions were dependent on values contained in a latitude-longitude datafile read into the model in subroutine inemit, the emission rate in the surface layer would be calculated in a loop such as:

```
      do 100 jl=1, jpnl
      .......
      factor(jl) = function of T, P, total number density, gridbox area
      emr(jl, jpspec) = data(jl, latitude, jpspec) * factor
100   continue
```

where the factor is used to convert the rate to molecules $cm^{-3}s^{-1}$, the units used by ASAD.

7.11.1 How to handle emissions and families

Getting the right effect on your chemistry when trying to do emissions when families are being used can be a bit of a problem. For example, let's consider the emission of NO from the surface. For a model in NO is being treated as a separate and independent tracer there's no problem. You supply a subroutine which sets the emission rate for NO. However, if NO is being used in a family, NO_x, then you have a problem because now NO is being considered as steady state. Including the emissions in the ratio calculation of the family and in the tendency can produce some odd results.

What you really want to do is simulate the emission of NO as NO_x, since this is the tracer that you are advecting. However, if you apply the emission rate of NO to the production of NO_x then you are producing too much odd oxygen, since NO_x = NO + NO_2 + NO_3. The method that we use to correct this is to apply a correction to the odd oxygen family, O_x, the same time as applying the emission of NO to NO_x. So, for example, if the change in NO_x should be ΔNO_x, then

$$\Delta O(^1D) = -\left(\frac{NO_2}{NO_x}\right)\Delta NO_x \qquad \text{(EQ 15)}$$

where we use the reaction

$$NO_2 + h\nu \rightarrow NO + O(^1D) \qquad \text{(EQ 16)}$$

In other words, we scale the increase in NO_x by the ratio of NO_2 to NO_x in the family. Strictly speaking we should include another term for the NO_3 but this is very small and we neglect it here. The ratio of NO_2 to NO_x is stored in ASAD or can be calculated at any point (see the routine ftoy).

Similarly, if we wanted to emit a source of NO_2 rather than NO we can use the same approach. In this case however, by adding a term to the NO_x we would be emitting too little odd oxygen; since some of our NO_2 emitted is really going to give increases in the rest of the family members, NO and NO_3. In this case, the correction in Eq. 15 is positive (no minus sign) and the ratio is NO to NO_x.

The next question is how best to apply this correction. There are two ways of doing it; either inside ASAD or outside. ASAD computes the total tendency of NO_x and will add contributions from the emissions and deposition terms (if they are being used). The first method then would add these corrections described above as additional tendency terms (see section 5.3 and on) using user supplied subroutines. The second method is to take a process split approach. Consider the tendency equation that determines the rate of a species, *y*.

$$\frac{dy}{dt} = P + L + E \qquad \text{(EQ 17)}$$

where *P* is the production, *L* is the loss rate and *E* represents the production due to emissions. We can rewrite this in two stages:

$$\frac{y^{n+1} - y^{n}}{\Delta t} = P + L + E \quad \text{(EQ 18)}$$

$$\tilde{y}^{n+1} = y^{n} + \Delta t(P + L) \quad \text{(EQ 19)}$$

$$y^{n+1} = \tilde{y}^{n+1} + \Delta t E \quad \text{(EQ 20)}$$

In the first method, you compute *E* and ASAD computes the new values of *y* via Eq. 18. In the second method, using a process split approach, you don't tell ASAD that emissions are being used at all and ASAD computes the updates value of *y* using Eq. 19. You then, in the calling model, after the call to CDRIVE, compute the emission rate and do a forward timestep using Eq. 20. Remember that if you are using families you will have to apply the emission rate to the family and possibly a correction as discussed above.

8 Acknowledgements

We are grateful to Lida Nejad who wrote the DELOAD program that originally inspired this work. We would also like to thank Oliver Wild, Jamie Kettleborough, Peter Stott and David Lary for useful discussions. Thanks also to Helen Rogers for the work on how to do emissions when using families. Sarah Goodchild contributed significantly to the testing of the code, but cannot be blamed for any remaining bugs, for which the little people are alone responsible. PDB acknowledges the support of the DoE. PDB is very grateful to The University of Strathclyde for use of computing facilities.

9 References

Atkinson R., D.L. Baulch, R.A. Cox, R.F. Hampson Jr, J.A. Kerr and J. Troe, 1992, Evaluated kinetic and photochemical data for atmospheric chemistry: supplement IV, *Atm. Env.*, **26A**, 1187.

Austin, J, 1991, On the explicit versus family solution of the fully diurnal photochemical equations of the stratosphere, *J. Geophys. Res.*, **96**, 12941-12974, 1991

Carver G.D., P.D. Brown and O. Wild, 1997, The ASAD atmospheric chemistry integration package and chemical reaction database, accepted for publication in *Comp. Phys. Comm.*

Carver G.D. and P.A. Stott, The IMPACT implicit time integration scheme, submitted to *Ann. Geophysicae*.

DeMore et al., 1992, Chemical kinetics and photochemical data for use in stratospheric modelling, Evaluation No. 10, Jet Propulsion Laboratory, Pasedena, Calif., Publication 92-20.

Hesstwedt E., Ø. Hov and I.S.A. Isaksen, 1978, Quasi-steady-state approximations in air pollution modelling, Int. J. chem. Kinet. 10 (1978) 971.

NAG, The NAG Fortran Library Manual, NAG Ltd, Wilkinson House, Jordan Hill Road, Oxford, UK (1996) D02 Chapter.

Brown P.N., G. D. Byrne, and A. C. Hindmarsh, 1989, VODE: A Variable Coefficient ODE Solver, SIAM J. Sci. Stat. Comput., 10, 1038.

Stott, P.A. and R.S. Harwood, 1993, An implicit time-stepping scheme for chemical species in a global atmospheric circulation model, *Ann. Geophysicae*, **11**, 377-388.

Millar, T.J. et al., 1991: “[UMIST database]”.

Nejad, L.A.M., 1986: PhD. Thesis, University of Manchester. 22.

Wayne R.P., 1991, Chemistry of Atmospheres, 2nd Edition, Clarendon Press, Oxford, p. 103.

Appendicies

A.1 License information

Users requesting the ASAD package from the Computer Physics Communications program library (the preferred method) will be asked to sign a non-profit use agreement. In addition, the authors provide the following license to use ASAD.

The ASAD package including all ASAD source code, documentation, utilities and UGAMP/ACMSU reaction database are distributed free of any charge but without any warranty, implied or otherwise, of any kind. The ASAD package is copyright Centre for Atmospheric Science, Cambridge University, 1995-97. We retain the copyright and all other legal rights to the package and make it available non-exclusively. All users must ensure that any and all copyright notices contained in the package remain at all times.

You are allowed to use, copy, distribute and modify the software (in source and binary forms) under the following terms and conditions. These conditions are designed not to be restrictive but to protect the rights of the developers and users. Any documentation, announcements and other materials related to use should acknowledge that the software was developed at the Centre for Atmospheric Science, Cambridge University.

1. This license applies to the ASAD ‘package’ which we define to be all the files in the ASAD distribution file (‘tarfile’) and includes this notice, all ASAD source code, all utility source code, the ASAD reaction database and the documentation. This license also covers the ASAD code when it is included in any ‘work’ or program or any code derived from the program. The originating ASAD package is that held and distributed by the Centre for Atmospheric Science. This license may be changed for future versions of the ASAD package.
2. You may freely copy and distribute the ASAD ‘package’ subject to the following conditions.
   (a) The ‘package’ must be distributed complete with all source code, documentation and with this license file and copyright notice(s) intact.
   (b) The authors wish to maintain a list of who has the package so we ask (but do not make it a condition of usage) that you contact them to inform them of who you have passed the package on to.
   (c) You may not charge for distributing the software nor offer a warranty of any kind.
3. You are free to modify the software in any way you choose provided:
   (a) The developers wish to maintain the ‘master’ version of the ASAD package. It is not a condition for usage but the developers would request that any changes which may be of scientific value be communicated to them so that it may be included in the originating source. Appropriate acknowledgement would be given for such changes.

(b) The modified package, if distributed beyond the users local group, must prominently state that code which has been modified so that any damage (of any kind whatsoever, including loss of data or inaccurate data) due to these modification does NOT reflect on the original ASAD package distributed by the developers.

4. You are free to use this software in any way you see fit and incorporate it or parts of it into your own programs provided:

   (a) You respect the research development effort for the package and make appropriate acknowledgement for the contribution ASAD made to your work.

   (b) You do not attempt to patent the ASAD package or any part of it.

   (c) You do not attempt to include the ASAD package or any part of it into a commercial product without obtaining prior written permission from the developers.

   (d) If the ASAD code or any part of it is included in your own programs any copyright notice within that code remains intact.

   (e) Neither the name of the authors, the Centre for Atmospheric Science nor Cambridge University may be used to endorse or promote products derived from the package without prior written permission.

5. You are not required to accept this license since you have not signed it. However, nothing else grants you permission to use, modify or distribute the ‘package’. Therefore by using, modifying or distributing the ‘package’ you indicate your acceptance of this license and all terms and conditions in it.

6. As this ‘package’ is distributed free of charge, there is no warranty for the package of any kind, implied or otherwise. The package is provided ‘as is’. The entire risk as to the quality and performance of the program is with the user.

7. In no event will the developers or Cambridge University be held responsible for any damages, of any kind whatsoever, (including loss of data or data being rendered inaccurate) arising out of the use or inability to use the ASAD package.

A.2 Utilities supplied with ASAD

This is a brief description of the utility programs supplied with ASAD in the utils/ subdirectory. The select program is covered in more detail elsewhere in this document.

select.f	program that takes the user defined set of species and selects the reactions from the input rate files.
sel_job	unix shell script to run the select program. Take a copy of select.f and sel_job; edit sel_job for your particular needs and then type ‘sel_job’.
mergerate.f	program to take a old rate file and update the rates for each reaction from a new ratefile with the updated rates going to a new ratefile. e.g. the new ratefile might be a new release of bimol.d. This program will allow you to change the updated

	rates without the need for editing. See program source for more comments at top of file.
mergerate	Executable program compiled for Suns.
f2up.c	C program that takes f77 source and changes it for the Cray update/nupdate program. Files *.f are copied to *.up with *deck statements added and ‘include’ statements changed to ‘*call’. Files containing common blocks are copied from *.h to *.cdk with the line ‘*comdeck’ added at the top of the file.
f2up	Executable program compiled for Suns.

A.3 Computational efficiency and implementing ASAD on multiple processors

Glenn has spent a lot of time with the ASAD code optimising it, mainly on the Cray YMP at the Rutherford Appleton Laboratory but also on the Suns and Silicon Graphics workstations in the group. A brief comment on the efficiency of the code and implementing it on different computers is probably warranted.

ASAD is written in standard fortran77. Our only extension is the use of 8 character names. We don’t believe this will present a problem for most modern day compilers. We have successfully compiled and run the code on a Sun Sparc workstation running SunOS 4.1.3 using Sun’s own FORTRAN compiler, a Silicon Graphics Indigo2 running IRIX 5.2 with SGI’s own compiler, an Apple MacIntosh running System 7.5 using the FORTRAN compiler from Fortner Research and a Cray YMP and J90 running UNICOS 6 with Cray’s cf77 compiler. Our colleagues around the world have also run ASAD on a Dec Alpha and a PC Pentium running Linux.

A Cray and a workstation have very different performance characteristics. What works well (in the code) for a Cray doesn’t work well on a workstation. Let’s consider the Cray (or vector machines in general) first.

All of the ASAD loops will vectorise on the Cray. Glenn used the ‘flowview’ command to analyse and tune the performance of the code. All the IF statements inside DO loops will vectorise. The code (particularly in PRLS) has been carefully written, tested and rewritten, to achieve the best performance with the Cray compiler. As an example, for our stratospheric chemistry using ASAD, we get a very respectable 100Mflops average from ASAD with peak at 180Mflops (on a single processor YMP) using vectors several hundred elements long. We have NOT used ASAD on multiple processors as yet but this is coming soon.

The key to getting the best out of a vector computer is to have long loops with lots of floating point calculation in them which don’t use lots of arrays but mix a small number of arrays with local variables. As ASAD uses a lot of indexing for the main number crunching it wasn’t always possible to have lots of floating point code inside the loops, although outer loop unrolling has been used to improve this. So, the best way to get top performance out of ASAD is to use vectors which are as long as possible. For example, when we started to run the stratospheric chemistry code

developed using ASAD we vectorized over levels (about 12) as this made the code for the deposition of condensed particles easier. However, Glenn altered the code so that ASAD was passed a complete latitude-height slice of the 3D model (typically about 1000 elements) and this resulted in a speed-up of a factor of 3 from vectorizing over levels. As the chemistry on each point proceeds independent of the other it is easy to do this. The only exception is in the heterogeneous and photolysis code which works in terms of levels. We therefore had to make some modifications to extract the levels from the slice passed to ASAD.

For a workstation however, the best performance comes typically from using small vectors. The reason for this is because of the small, fast cache memory attached to the processor. You will get the best performance if you can keep all the variables in this cache. In terms of ASAD, this means using small loops or passing ASAD a single level of part of a level.

This is the reason why the parameter JPNL is not named ‘level’ or ‘longitude’ as it can be used for any spatial dimension or combination. Or, if you were running ASAD along a trajectory model, JPNL would be the total number of trajectories that you were using. If you were really keen you could change the size of JPNL and plot a curve of CPU time against length of vector for different workstations.

A.4 Programming standard

We originally wrote the package in the hope it would be useful to other groups developing atmospheric chemistry packages as well as to other UGAMP members. We have tried to write the code in a modular fashion with adherence to UGAMP FORTRAN programming standards. We apologise for the use of lower case but UNIX and upper case don’t mix (especially when using ‘vi’ eh Paul?).

We strongly encourage anyone who modifies or adds to the ASAD code to follow our example and stick to the programming standards and style found in the code and summarised below. If anyone wants further details, a UGAMP Internal Report describing the programming standard can be sent to them. What follows is a brief summary of that report.

The UGAMP programming standard is based on one used at the European Centre for Medium Range Weather Forecasts (ECMWF). It requires that code should be well documented and variables defined so that their scope and use is evident from their name.

A.4.1 Subroutine structure

Each subroutine should start with a comment block with the following sections: Purpose: a brief description of what the subroutine does. Interface: how the subroutine is called, what the arguments are and how they are altered. Method: how the subroutine does what it does. This block should refer to the relevant sections of the code if necessary (see below). Externals: what other subroutines or functions are

called. Modifications: how the subroutine has been modified. Local variables: (optional) a list of key local variables and what they are used for. If any variables are used in a SAVE statements, this comment block must be present and contain a description of these variables.

Then in the subroutine itself, the code should be separated into sections and subsections (much like a document would be) where each section has a distinct purpose. At the head of each of these sections is a title describing briefly what the section does. Each section should be separated from the previous by a line of dashes. Subsections should not be so separated.

All labels in a section should begin with the section number they are contained in. For example a DO loop in section 2.1 should be numbered 200, 210, 220 etc.

All COMMON blocks should be named starting with the letters ‘COM’ or ‘CM’. At the head of each block, there should be a brief title and description of what the block is used for. There should also be a comment block which has a list of the variables in it and what they are used for. Default values should also be indicated.

A.4.2 Variable naming convention

ECMWF and UGAMP use a naming convention for FORTRAN variables. We have found this to be very useful during development and debugging since it makes the scope and use of the variable readily apparent. We encourage anyone writing code for ASAD to follow this convention. There’s probably some places in the code where we didn’t follow the convention but we’ll gradually weed them out!

This table summarizes the variable naming convention. The first two letters are reserved, the remaining letters should indicate the variable usage. We allow the use of 8 characters in a variable name.

Table 6:

	Integer	Real	Logical
COMMON	M or N	A to Z	L
Parameter	JP		
Argument	K	P	O
Local	I or J	Z	G
Loop	J		

A.5 Species naming convention

Species naming convention and nomenclature - courtesy of Oliver Wild, Department of Chemistry, University of Cambridge.

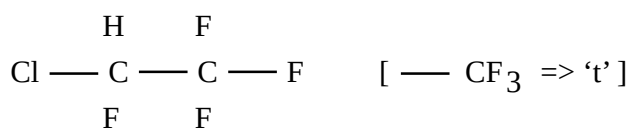
Inorganics	-ONO	is used for most NO2
	-ONO2	for most NO3
	-O2NO2	for most NO4
	-OO	for most peroxides
Organics	Me	for methyl CH3-
	Et	for ethyl CH3CH2-
	Pr	for propyl C3H7- (i- iso; n- normal)
	Bu	for butyl C4H9 (i, n, t)
	-CHO	for aldehydes

The **halocarbons** *are written as a letter followed by a number and possibly another letter (the number being the standard notation for CFCs), as follows:*

F	freon (halon with 1 carbon atom assumed)
H	other halons
R	halocarbon radical, carbon atom activated
first digit:	number of C atoms less 1
second:	number of H atoms plus 1
third:	number of F atoms
fourth:	Cl atoms
fifth:	Br atoms
t	if the terminal group is -CF3
d	if the terminal group is -CF2A
m	if the terminal group is -CFAB

For example:

H1241t (C_2HF_4Cl)



A.6 Check list for adding a new integrator

This is a list of the code in ASAD that needs changing when a new integrator is added.

1. Edit CDRIVE to add the call to the new integrator. The convention in ASAD is that all integrators use the phrase ‘DRIVE’ or ‘DRIV’ in the subroutine name.
   (a) If the method is a stiff integrator, choose a value of METHOD .GE. 10.
   (a) If the method is not a stiff integrator, choose the next highest value .LT. 10.
2. Edit CMCCTL and add the new integrator to the description of the variable METHOD. Remember to add some comments to the description of METHOD.

3. Add to the comments in CINIT to include new integrator in list for KMETHOD argument.
4. If integrator requires variables to be set by the user:
   (a) Create a new COMMON block to hold these variables.
   (b) Assign the position of the new variables in the switch arrays passed to CINIT.
   (c) Document these new arguments in the comment block to CINIT.
   (d) Put code in CINIT to assign COMMON block value and check the allowed values of the passed variable.

A.7 Known bugs and problems

As these will change more often than this document, you should always be able to find a file called BUGS included with the ASAD distribution. If not, please contact the maintainer of the package for a list.

If anyone spots any mistakes or omissions in this user guide please let us know!

Index

2D model 36
3D model 36

A
Absolute tolerance 22
Adams method 23
Adding a new scheme to ASAD 50
Apple MacIntosh 4, 32
  MPW 32
Aqueous phase chemistry 40
Arrhenius expression 9
ASAD
  acronym 1
  adding a new integrator 50
  available from 2
  bugs 2, 51
  calculation of family ratios 22
  compiling 3
  constant field type 7
  distribution of software 1, 45
  editing 29
  family type 6
  heterogeneous ratefile 3
  how species are treated in 5
  info.doc 12
  initialising 19, 38
  input files 17
  interface to calling model 16
  license 2, 45
  list of species 3, 30
  master ratefiles 2
  modifying parameter 18, 29
  parameters controlling 17, 25
  photolysis ratefile 3
  rates subdirectory 2
  renumbering ratefiles 3
  source code 2
  species names 5
  steady state species type 6
  tests 2
  tracer type 6
  tracer/family type FT 6
  unpacking 2
  use on vector machines 16
  using on an Apple MacIntosh 32
  utility programs 2, 3
  version history 2
  world-wide-web home page 2, 29
Species
  as defined by ASAD 3
Atkinson et al. 8
ATOL 22
Austin 5, 20

B
BDF time integration scheme 19
BIMOL 9
Bimolecular ratefile 2, 9
  exceptions to Arrhenius expression 9
Box model 16, 24, 36
Bugs and known problems 2, 51

C
Carver and Stott 16, 23
Carver et al. 15
cdot 20, 39
CDRIVE 17, 20, 26, 27, 39
  arguments to 20
CF 25
chch.d file 5
  constant field type 7, 25, 27
  constant species type CT 6
  editing 33
  example of 30, 31
  family type 6
  format of file 5
  steady state species type SS 6
  tracer type 6
  tracer/family type FT 6, 34
  use of flags in 34
Chemical families 5, 20
  major member 21
  member of 6
  minor member 21
  odd atoms 6
  odd oxygen 21
  passed from calling model 20
  tracer/family type FT 6
  use of 16
Chemical lifetime 16
Chemical loss 15
Chemical production 15
Chemical tendencies 20
Chemistry timestep 19, 38
CINIT 17, 19, 26, 27, 34, 38
  arguments to 19
Closed reaction set 12, 31
cmdpem 26, 27, 40
cmphys 25
cmreac 26, 27
cmspec 34
cominx 26
Computer Physics Communications 2, 45
Constant field type 7, 25, 27
Constant species 6
Convergence of iteration scheme 22
Convergence test 37
Cray 4
Cray J90 4
Cray YMP 4

D
D02EAF 23
DELOAD 43
DeMore et al. 8
Deposition rates 15, 27
Deposition scheme 40
Documentation
  userguide 2
dpd 27, 40
dpw 27, 40
Dry deposition 19, 40
  setting the rates 40
DRYDEP 26, 27, 40
dtime 20, 38

Dummy subroutines 26
Dynamical timestep 16

E
ECMWF 48
Editing ratefiles 12, 36
  removing species 13
Email
  Computer Physics Communications program library 2
Emissions 15, 20, 40
  setting the rates 40
  when using families 41
EMISSN 26, 40
emr 26, 40

F
F2UP 3, 47
Families
  See Chemical families 5
First guess
  See IMPACT
Fractional products 13
ftol 37
FTOY 19, 22, 37
ftr 20, 39
FYINIT 34

H
Hesstwedt et al 23
HETERO 16, 27, 40
Heterogeneous ratefile 3, 11, 27
  omitting heterogeneous reactions 11
Heterogeneous rates 27
  setting them 40
  turning off 19
Heterogeneous scheme 40
High pressure limit 10

I
IMPACT 16, 19, 23, 37
  first guess 38
  Newton-Rhapson iteration 19, 38
INDDEP 16, 26, 40
INDEPN 26
Indexing rate array 25
Indigo 4
INEMIT 16, 26, 40
info.doc file 12, 34, 37
  example of 32
INICNT 27
Initialising ASAD 38
INPHOT 16, 26, 35, 40
Input files 17
Interchange reactions 22
INWDEP 16, 26, 40
Iteration
  convergence in ftoy 22
  convergence test 37
Iteration scheme
  computing family ratios 22
IUPAC 8

J
jpbk 18, 37
jpctr 17, 37
jphk 18, 37
JPL 8
jpnl 18, 25, 37
jppj 18, 37
jpspec 18, 25, 37
jptk 18, 37

K
kcdt 19
kfphot 19
Kinetic data
  IUPAC assessment 8
  JPL reaction handbook 8
kmethod 19

L
LAPACK 18
ldepd 19, 26
ldepw 20, 26
lemit 20
lhet 19, 27
License 2, 45
Linux 4
Loss terms 21
Low pressure limit 10
lvmr 19, 21, 39

M
Major family member 21
makefile 2
Mergerate 3, 13, 46, 47
Methane oxidation 30
Minor family member 21
Model
  2D 36
  3D 36
  calling ASAD 39
  how to set it up for ASAD 36
  initialising 26
  initialising tracers in 32
  tracers used by 34, 37
  transport component 15
MPW 32

N
NAG 19, 23
ncdt 38
NETLIB 18, 19
Newton-Rhapson
  See IMPACT
nhrkx 26
nit0 19
nitfg 19, 38
nitnr 19, 38
nprkx 25
nrsteps 19, 38
Number density 19, 21
Nupdate 2, 4

O
Odd atoms 6
Odd oxygen 21
oemit 26
Open .v. closed reaction set 35
Open reaction set 12, 31

P
paramc 18, 25, 29, 37

Parameters
changing 18, 29
example of setting them 37
use of 17
Pentium 4
peps 18
PHOTOL 16, 26, 39
Photolysis 3
frequency of rate calculation 19
Photolysis ratefile 11
building your own 34
Photolysis scheme 39
initialising 40
Physical variables 25
pmin 37
Polar stratospheric clouds 40
Pressure 20, 25, 39
Production terms 21
ptol 37

Q
QSSA
Quasi-steady-state time scheme 19, 23

R
Ratefiles
bimolecular ratefile 2, 8, 9
comments in 8
compiling 8
customising 8
editing them 12, 34
example of editing 36
excluding upper limit reactions 12
heterogeneous ratefile 8, 27
master 2
open, closed reaction set 12
photolysis ratefile 8
reactions with multiple branches 10
removing species 13
renumbering 33
specifying fractional products 13
subroutine to read them 35
trimolecular ratefile 3, 8, 10
unknown products 8
updating master ratefiles 13
Rates 2
indexing rate array correctly 25
rate array 25
rath.d 27, 40
Ratios 22
ratj.d 26
Reactions
deleting upper limit 36
family interchange 22
reordering them 25
with multiple branches 36
README 2
READRAT 35
Relative tolerance 22
Removing species 33
RENUMB 3, 13, 33, 36
Renumbering ratefiles 33
rk 25
RS6000 4
RTOL 22

S
Saving CPU time 33
sel_job 11, 31, 46
format of job script 11
SELECT 3, 5, 11, 31, 46
Short lived species 21
Silicon Graphics 4
slamch 18
Solaris 4
Source code 2
SPARCstation 4
Species
concentrations bounded by ratios 22
defining as a constant field 7, 25, 27
defining as constant 6
long lived 21
names 5
nomenclature 5
removing from chch.d file 33
short lived 21
steady state 21, 22
tracers 6
treating as family members and tracers 23
species.d 30
Steady state species 6, 21, 22
Stott and Harwood 16
Stratosphere 34
SVODE 19, 23

T
Temperature 20, 25, 39
Termolecular ratefile 3, 10
exceptions to general expression 11
Tests 2
Time integration 50
Adams method 23
adding more schemes to ASAD 16
choice of method 19
conservation of 23
first guess 19
IMPACT scheme 23
NAG BDF method 19, 23
non-stiff 19, 23
QSSA 19, 23
stiff 19, 23
SVODE 19, 23
timestep of calling model 20
use of schemes during development 24
vectorisation issues 24
Timestep 20
determining 38
TND 21
Total number density 25
Tracers 6
as families or species 5
initialising correctly 32
passed from calling model 20
Tracers in model 34
Trajectory models 16
Troposphere 34
Tropospheric chemistry 30

U
UGAMP 4
Unicos 4
Unimolecular reactions 10
Unix

redirecting input 32
tarfile 2
Unknown products
deleting reactions with 36
UPDATE 4
Upper limits 12, 36
URL
ASAD home page 2
Computers Physics Communications program library 2
User supplied subroutines 25
Utilities 2, 46
directory 3
F2UP 3
MERGERATE 3
RENUMB 3
sel_job 3
SELECT 3

V
Vectorisation 16, 36, 39
Versions 2
Volume mixing ratio 19

W
Wayne 10
Wet deposition 20, 40
setting the rates 40
WETDEP 26, 27, 40
World-Wide-Web
ASAD home page 2, 29
Computer Physics Communications program library 2

Z
Zero divide
guarding against 18